AIMS

Arbeitspapiere des Instituts für Maschinelle Sprachverarbeitung
Lehrstuhl für Theoretische Computerlinguistik Universität Stuttgart

1998 • VOL. 4 • NO. 3

Inducing Lexicons with the EM Algorithm

Previous Issues of AIMS:

Vol.1 (1) 1994: *Wigner Distribution in Speech Research.* Master thesis by Wolfgang Wokurek (in German), papers by Grzegorz Dogil, Wolfgang Wokurek, and Krysztof Marasek (in English), and a bibliography.

Vol.2 (1) 1995: *Sprachentstörung.* Doctoral Dissertation. University of Vienna, 1994 by Wolgang Wokurek (in German with abstract in English). Full title: Sprachentstörung unter Verwendung eines Lautklassendetektors (Speech enhancement using a sound class detector).

Vol.2 (2) 1995: *Word Stress.* Master thesis by Stefan Rapp (in German) and papers mostly by Grzegorz Dogil, Michael Jessen, and Gabriele Scharf (in English).

Vol.2 (3) 1995: *Language and Speech Pathology.* Master theses by Gabriele Scharf and by Jörg Mayer (in German) and papers by Hermann Ackermann, Ingo Hertrich, Jürgen Konczak, and Jörg Mayer (mostly in German).

Vol.3 (1) 1997: *Tense versus Lax Obstruents in German.* Revised and expanded version of Ph.D. Dissertation, Cornell University, 1996 by Michael Jessen (in English). Full title: Phonetics and phonology of the tense and lax obstruents in German.

Vol.3 (2) 1997: *Electroglottographic Description of Voice Quality.* Habilitationsschrift, University of Stuttgart, 1997 by Krysztof Marasek (in English).

Vol.3 (3) 1997: *Aphasie und Kernbereiche der Grammatiktheorie (Aphasia and core domains in the theory of grammar).* Doctoral Dissertation, University of Stuttgart, 1997 by Annegret Bender (in German with abstract in English).

Vol.3 (4) 1997: *Intonation and Bedeutung (Intonation and meaning).* Doctoral Dissertation, University of Stuttgart, 1997 by Jörg Mayer (in German with abstract in English).

Vol.3 (5) 1997: *Koartikulation und glottale Transparenz (Coarticulation and glottal transparency).* Doctoral Dissertation, University of Bielefeld, 1997 by Kerstin Vollmer (in German with abstract in English).

Vol.3 (6) 1997: *Der TFS-Repräsentationsformalismus und seine Anwendung in der maschinellen Sprachverarbeitung (The TFS Representation Formalism and its Application to Natural Language Processing).* Doctoral Dissertation, University of Stuttgart, 1997 by Martin C. Emele (in German).

Vol.4 (1) 1998: *Automatisierte Erstellung von Korpora für die Prosodieforschung (Automated generation of corpora for prosody research).* Doctoral Dissertation, University of Stuttgart, 1998 by Stefan Rapp (in German with abstract in English).

Vol.4 (2) 1998: *Theoriebasierte Modellierung der deutschen Intonation für die Sprachsynthese (Theory-based modelling of German intonation for speech synthesis).* Doctoral Dissertation, University of Stuttgart, 1998 by Gregor Möhler (in German with abstract in English).

Inducings Lexicons with the EM Algorithm

Mats Rooth, Stefan Riezler, Detlef Prescher, Sabine Schulte im Walde, Glenn Carroll, and Franz Beil

Lehrstuhl für Theoretische Computerlinguistik
Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Rooth, Mats:
Inducing Lexicons with the EM Algorithm / Mats Rooth; Stefan Riezler; Detlef Prescher; Sabine Schulte im Walde; Glenn Carroll; Franz Beil. - Stuttgart, 1998.
AIMS - Arbeitspapiere des Instituts für Maschinelle Sprachverarbeitung, Vol. 4, No. 3, 1998, Stuttgart, Germany.
ISSN 1434-0763

Lehrstuhl für Theoretische Computerlinguistik
Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
Azenbergstraße 12
70174 Stuttgart

www: `http://ww.ims.uni-stuttgart.de/projekte/gramotron/`
email: `gramotron@ims.uni-stuttgart.de`

Contents

Two-dimensional Clusters in Grammatical Relations *

Mats Rooth

* Revised version of paper presented at *Symposium on Representation and Acquisition of Lexical Knowledge. Polysemy, Ambiguity, and Generativity.* AAAI 1995 Spring Symposium Series, Stanford University, 1995.

1.1 Introduction

If someone asked me to explain the word "product" as used below, I might say that it refers to things like drugs, cars, and software which are developed, produced, sold, and used.

(1) The company announced a new product.

While it is drugs, cars and software which are serving as examples of products, the list of verbs has a role as wel. If I merely said that products were things like drugs, cars, and software, someone would hardly be in a position to say whether toothpaste counts as a product. And a definition of "product" as simply "something which is produced" gets a more general sense of the word, including things such as toxic waste produced by an industrial process, and carbon dioxide produced by human respiration.

The explanation of "product" by multiplied example involves the implicit claim that drugs, cars, and software are all developed, and all produced, all sold, and all used. Not only is this true, but in a large enough text corpus we can find examples examples of drugs, cars, and software being *described* as being developed, produced, sold and used, in all combinations:

(2) a. 19516892 Their goal is to develop new drugs to compete in world markets .
b. 44435981 Genzyme sells specialty chemicals used by other companies to produce drugs , diagnostic tests and other products and performs contract research for drug companies .
c. 44222175 " We absolutely must be in the U.S. market by 1992 , " he says , " to sell the new drugs our research labs will be producing . "
d. 35971334 Milton Bass , a New York attorney for Zenith , asserted that the appeals court's ruling will spur civil suits against other drug companies that have " conducted campaigns to frighten doctors and others not to use generic drugs . "

(3) a. 56955552 The Europeans' problem: the huge sums it takes to develop new cars and bring them to market .
b. 47029320 The eight major auto makers didn't produce any cars last week because of the holidays .
c. 17150902 " This year , for the first time , we've had to work to sell the cars , " says Michael J. Jackson , a Saab dealer in suburban Washington , D.C .
d. 61132866 – A halt to imports of sedans , plus a requirement that top officials use only Chinese-made cars .

(4) a. 12402911 " This settlement paves the way for the competition to develop software in the operating system arena . "

b. 53998616 He guessed it will take Microsoft six to 12 months to produce the software based on the Sybase technology .

c. 34138820 The company operates retail stores that sell computer software .

d. 17518485 They are writing a book to show campaign managers how to use Lotus 1-2-3 software to analyze local voting habits .

These examples are from a corpus of Wall Street Journal newspaper stories, containing roughly 60 million word-like tokens (Liberman 1992). The number at the beginning of an example indicates its position in the corpus.

This combinatory pattern evident in these sentences is of independent interest, because it suggests a simple way of capturing selectional patterns relating verbs to their objects, or lexical pairs participating in other grammatical relations. In a number of problems arising in computational linguistics, we need to be able to decide whether a given phrase is an appropriate filler for a grammatical relation assigned by another word. This often comes down to a question of semantic compatibility between the head of the filler phrase and the semantic role associated with the grammatical relation. For instance, in the sentence below, *resulted* and *approved* both have morphological readings as both tensed verbs and past participles.

(5) The charge resulted from a settlement approved yesterday.

The morphological indeterminacy results in syntactic ambiguity. In the first structure in the first row of table 1.1 , *charge* is the subject of *resulted* and *approved* a postmodifier of *settlement.* In the second analysis, *resulted* is a postmodifier of *charge*, and *charge* is the subject of *approved.* Deciding between the syntactic analyses below comes down, at least in part, to a matter of semantic selection. In choosing an analysis, it would be useful to know (among other things) whether *settlement* is a good object for *approve* (it is) and whether *charge* is a good subject for *approve* (it is not). (Note that in terms of the underlying role assigned, the postmodified noun phrases are equivalent to objects of their modifiers.) The sentence below has the same ambiguity, and in this case it is the second syntactic analyis which is correct.

(6) Private-sector union contracts signed in the third quarter granted slightly lower wage increases.

Suppose that we wanted to attack such problems using information from a training corpus. Since thousands of verbs and nouns are involved, we can not conclude that a given verb-object pair is impossible, simply because we can not find it in the corpus. Take for instance a slightly more uncommon product verb such as *export*, and a more uncommon product noun, such as *engine.* Although the verb-object pair *export engine* is intuitively plausible, it was not detected at all by the method described below in a six million word training corpsus. A selection model which generalizes among words has the potential of solving the problem, since while the verb

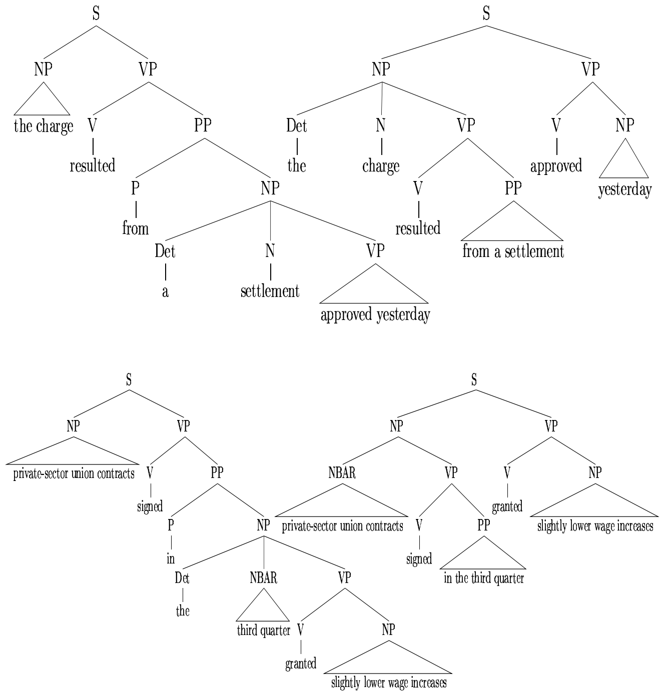


Table 1.1: First row: corrext and incorrect syntactic analyses for sentence 5. Second row: incorrect and correct syntactic analyses for sentence 6.

	asset	average	bit	bond	cent	cost	debt	dividend	foot	interest	mark	pence	point	price	rate	rating	security	share	stake	stock	tax	unit	value	yen
acquire	16		1			1	2			35							5	77	87	29		19		
boost		1		1		2	1	18		1	1		1	39	21	5		28	59	10	1		19	1
buy	16	2	2	48			2			53	1						36	348	107	190		29		2
climb					8	1					2	10	13	1										
cut						104	10	11					1	66	64	11			5	2	30		5	
decline		2	3		18	2	1						13	9	16			1		3				
drop	1	1	1		19				2		1	2	30	9	6	5								7
dump	1			3													2	10		10				
fall		2	1		132				1		2	14	171	38	33					1				28
gain					20					3	2	11	62		9			25	4		25			28
hold	18			7	1				1	22				3			5	68	121	30		3	1	
increase	6	3			2	25	5	26		8		1	3	26	36	2	1	36	75	2	11		16	
jump		2			3						2	8	9											
lower						20	2			1				23	83	55			16	1	2		4	
plunge					3						2		14	4						2				
purchase	8			5			2		1	17							6	95	24	20		6		
push	1	2		2		1				1			3	44	20		1	4		16	1	1	2	1
raise						23	5	28		8	5			131	149	26		5	74		46		11	1
reduce	9	1			1	76	105	3		5			1	22	55	8		9	41	2	26		21	
retain	1							1		13								17	21	1		3	1	
rise		13	9		136	18			2	2	3	52	125	18	19			1		1		1	2	22
sell	114	2	1	40			6			72	2		1	12	8		48	243	144	149		104		2
slash						17	4	9						20	6				1	1	3		3	
trade	1	1		2	2	2								9			7	22	2	37		5		1

Table 1.2: Frequency counts for 24 verbs and 24 object heads.

export *export* does not occur with *engine*, it occurs with other product nouns, and *engine* occurs with other product verbs.

The purpose of this paper is to develop a mathematical and computational model which captures the notion of a selectional dependency between a set of verbs and a set of nouns, or more generally two sets of words participating in a grammatical relation. Section 2 describes a simple categorical selection model, which in section 3 is given a probabilistic twist. Locally optimal probabilistic models can be generated by an incremental procedure similar to Baum-Welch reestimation of hidden Markov models. In section 4 we look at results for a reasonably large sample of verbs an nouns. Section 5 discusses applications of the probabilistic model, giving preliminary results for a parsing problem.

1.2 Categorical selection types

Table (1.2) is a matrix of frequency counts for verb-object occurrences of twenty-four verbs and twenty-four nouns. The table is a sub-part of a verb-object matrix derived from an approximately 6 million word sample of the Wall Street Journal, parsed by Donald Hindle with his Fidditch parser.[1] I extracted verb-noun pairs with a Lisp program from list representations of parses, and mapped them to uninflected forms using a full form word list. The resulting list contained 84182 non-zero frequency counts. The frequency matrix was reduced by eliminating rows (corresponding to verbs) with fewer than five non-zero entries, and subsequently eliminating columns with fewer than five non-zero entries. This gave a matrix indexed by 992 verbs and 1027 nouns, containing 55251 non-zero entries. The verbs and nouns in the 24 × 24 sub-table were selected by hand for illustrative purposes.

[1] The parser is described in Hindle (1983) and Hindle (1994). Similar verb-object data is discussed in Church et al. (1991).

	asset	bond	interest	security	share	stake	stock	unit	average	bit	cent	foot	mark	pence	point	yen	cost	debt	dividend	price	rate	rating	tax	value
acquire	16		35	5	77	87	29	19		1							1	2						
buy	16	48	53	36	348	107	190	29	2	2			1			2		2						
dump	1	3		2	10		10																	
hold	18	7	22	5	68	121	30	3			1	1								3				1
purchase	8	5	17	6	95	24	20	6				1						2						
retain	1		13		17	21	1	3											1					1
sell	114	40	72	48	243	144	149	104	2	1			2		1	2		6		12	8			
trade	1	2		7	22	2	37	5	1		2					1	2			9				
climb											8		2	10	13		1			1				
decline					1		3		2	3	18				13		2	1		9	16			
drop	1								1	1	19	2	1	2	30	7				9	6	5		
fall							1		2	1	132	1	2	14	171	28				38	33			
gain			3		25	4					20		2	11	62	28					9		25	
jump									2		3		2	8	9									
rise			2		1		1	1	13	9	136	2	3	52	125	22	18			18	19			2
plunge							2				3		2		14					4				
boost		1	1		28	59	10		1				1		1	1	2	1	18	39	21	5	1	19
cut						5	2								1		104	10	11	66	64	11	30	5
increase	6		8	1	36	75	2		3		2			1	3		25	5	26	26	36	2	11	16
lower			1			16	1										20	2		23	83	55	2	4
push	1	2	1	1	4		16	1	2						3	1	1			44	20		1	2
raise			8		5	74							5			1	23	5	28	131	149	26	46	11
reduce	9		5		9	41	2		1		1				1		76	105	3	22	55	8	26	21
slash						1	1										17	4	9	20	6		3	3

Table 1.3: The same counts in another order.

Re-arranging the rows and colums in the small table brings out a dependency between the rows and columns. In table 1.3, most of the non-zero entries are in the three blocks on the diagonal. We can think of the block organization as capturing three semantic selectional types within this part of the verb-object grammatical relation. The first block corresponds to a notion of exchange of financial instruments or ownership interests. The second block involves measurement of a scalar motion by some dimensioned quantity, and the third block involves change in some scalar quantity, such as a stock price. To represent the blocks, we need not re-order the matrix: we can equate a block with a pair of a subset of the verb set and and a subset of the noun set. A closer examination shows that it is not reasonable to insist that the noun sets for different blocks be disjoint. The noun *stake* occurs freqently with the verbs of the third block (e.g. *boost, increase, raise, and reduce*), as well as with the verbs of the first (e.g. *acquire, buy, sell, and trade*). Here are some example sentences:

(7) a. 1727999 Most of Japan's big computer companies hold a 1% stake in Ascii, and Mitsui & Co., one of Japan's largest trading companies, plans to boost its stake to 5%.

b. 1266211 Though Koito is resisting, and Mr. Pickens has announced plans to increase his stake to 26%, he maintains "there's nothing hostile" about his investment.

c. 1690314 Ford will raise its Jaguar stake to the maximum 15% after the 30-day waiting period expires, the Ford executive said.

d. 2081122 The sales reduce the group's stake to 740,500 shares.

(8) a. 5786677 Galesi Group said it is seeking to acquire Lone Star Technologies Inc.'s stake in American Federal Bank for $58 million.

b. 1024554 The Italian financier is close to announcing that he's setting up a holding company to buy controlling stakes in Hungarian companies, according to Hungarian sources.

c. 2668376 Domino's Pizza owner Thomas Monaghan may sell his 97% stake in the chain, which is the nation's largest pizza delivery company.

d. 4287292 One possibility would be for the U.S. group, formally called Newgateway PLC, to trade its troublesome stake to Isosceles for certain Gateway assets.

The reason for the overlap is that a stake (say in a company) is both something which can be bought or sold, and a scalar quantity which can be increased or decreased.

The verb *increase* is in principle a symmetric example, though this is not really obvious in these data. It occurs both with objects denoting a changing scalar quantity, and with a objects (or pseodo-objects) measuring that change:

(9) a. 6664778 Jack W. Forrest, Environmental Systems president and chief executive officer, said the change increases the company's effective tax rate to about 35% from 20%.

b. 1891048 If Mr. Wanniski's theory is right, that a tax cut increases the value of capital assets held by owners, wouldn't that increase also represent a rather dramatic and totally unjustified inflation of those values?

c. 4102768 Georgia Gulf said it increased the exercise price of the rights to $120 from $50.

d. 3578950 "It has increased the level of caution in the market," said the Chicago trader.

(10) a. 3337850 Economists said the August civilian unemployment rate will have increased 0.1 percentage point to 5.3%.

b. 3854070 Unleaded-gasoline futures were mixed, although the September contract increased 0.22 cent to settle at 54.15 cents a gallon.

In the latter group, the changing scalar quantity is realized as subject.

Given two vocabularies V and N, we define a selection type as a pair $\langle V', N' \rangle$, where $V' \subseteq V$ and $N' \subseteq N$. A selectional model is simply a set of selectional types, and given what was said above, we should not impose any requirement of non-overlap for the noun sets or verb sets of a selection model. A reasonable selection model for the 24 × 24 frequency matrix

is:

$$\left\{ \begin{array}{l}
\left\langle \left\{ \begin{array}{l} \text{acquire buy} \\ \text{dump hold} \\ \text{purchase retain} \\ \text{sell} \end{array} \right\}, \left\{ \begin{array}{l} \text{asset bond} \\ \text{interest security} \\ \text{share stake stock} \\ \text{unit} \end{array} \right\} \right\rangle \\
\left\langle \left\{ \begin{array}{l} \text{climb decline} \\ \text{drop fall gain} \\ \text{jump rise plunge} \\ \text{increase} \end{array} \right\}, \left\{ \begin{array}{l} \text{average bit cent} \\ \text{foot mark pence} \\ \text{point yen} \end{array} \right\} \right\rangle \\
\left\langle \left\{ \begin{array}{l} \text{boost cut increase} \\ \text{lower push raise} \\ \text{reduce slash} \end{array} \right\}, \left\{ \begin{array}{l} \text{cost debt} \\ \text{dividend price} \\ \text{rate rating tax} \\ \text{value share stake} \\ \text{stock} \end{array} \right\} \right\rangle
\end{array} \right\}$$

1.3 Probabilistic selection types

The above construction, because it is categorical, encodes no information about the relative freqeuncy of, for instance, *buy* and *dump* as verbal realizations of the first selectional type. In applications, having access to graded information is useful, in that it allows the large numbers of analyses — such as syntactic parses — to be ranked. Futhermore, once we get beyond simple examples, it is not clear that membership in selectional patterns should be considered discrete.[2] Among ways of introducing graded distinctions, probabilistic models are appealing, because they have the potential of telling us, in complex situations, how a number of graded distinctions are to be combined. The simple recipe for turning a categorical model into a probability model is replace characteristic functions of sets with probability distributions. In the present case, we redefine a selection type as a pair of discrete distributions, one on the verbs and one on the nouns:[3]

$$\langle \lambda v p^{\tau}_{v}, \lambda n p^{\tau}_{n} \rangle \quad \textstyle\sum_{v \epsilon V} p^{\tau}_{v} = 1 \quad \textstyle\sum_{n \epsilon N} p^{\tau}_{n} = 1$$

The function $\lambda v p^{\tau}_{v}$ maps a verb to a number in the interval $[0, 1]$, meeting the constraint that the set of verb probalities sums to one, and similarly for the nouns. In order to use this notation, we must assume that the verb, noun and type sets are (or have been rendered) disjoint: we do not want to identify the verb probability $p^{\tau}_{\text{increase/V}}$ with the noun probability

[2] Furthermore, I did not say what makes one categorical selectional model better than another, and how they are to be discovered computationally. I have experimented with an incremental search for selection models, using a Solomonoff-Kolmogoroff-Chaitin measure to evaluate the combination of the complexity of the model with the complexity of describing the data matrix given the model. I will not describe this method here, since the search was computationally expensive, and the results only moderately encouraging.

[3] $\lambda v p^{\tau}_{v}$ generates a probability distribution $\lambda X \sum_{v \epsilon X} p^{\tau}_{v}$ measuring subsets of V. In the text, I surpress the distinction between discrete probability distributions and their generators.

type 1 .204				type 2 .315				type 3 .481			
fall	.344	point	.364	raise	.265	rate	.238	sell	.319	share	.340
rise	.343	cent	.281	reduce	.189	price	.207	buy	.288	stake	.198
gain	.128	pence	.080	cut	.162	cost	.143	hold	.095	stock	.172
drop	.059	yen	.071	lower	.109	stake	.108	acquire	.093	interest	.078
decline	.038	price	.060	increase	.100	debt	.071	purchase	.063	asset	.065
climb	.028	rate	.059	boost	.073	tax	.063	increase	.030	unit	.059
jump	.020	tax	.021	push	.036	rating	.059	trade	.027	security	.038
plunge	.019	average	.017	slash	.034	dividend	.050	boost	.024	bond	.037
increase	.006	cost	.016	sell	.010	value	.044	retain	.019	debt	.004
push	.005	bit	.011	decline	.009	interest	.006	gain	.011	price	.002
trade	.003	mark	.011	drop	.006	stock	.003	dump	.009	average	.002
reduce	.002	foot	.004	trade	.005	mark	.003	push	.009	yen	.002
boost	.001	stock	.002	hold	.002	average	.002	reduce	.008	bit	.001
sell	.001	value	.002	retain	.001	asset	.002	raise	.003	mark	.001
cut	.001	interest	.002	acquire	.001	yen	.001	decline	.001	foot	.001
hold	.001	unit	.001	gain	.000	share	.000	rise	.001	value	.000
raise	.000	asset	.000	rise	.000	point	.000	plunge	.001	rate	.000
buy	.000	share	.000	plunge	.000	cent	.000	drop	.000	point	.000
slash	.000	rating	.000	fall	.000	pence	.000	slash	.000	cost	.000
lower	.000	stake	.000	climb	.000	bond	.000	lower	.000	cent	.000
acquire	.000	debt	.000	purchase	.000	security	.000	cut	.000	dividend	.000
purchase	.000	dividend	.000	buy	.000	bit	.000	fall	.000	tax	.000
retain	.000	security	.000	jump	.000	unit	.000	jump	.000	pence	.000
dump	.000	bond	.000	dump	.000	foot	.000	climb	.000	rating	.000

Table 1.4: Parameters of a selection model.

$p^{\tau}_{\text{increase}/\text{N}}$. We also add a probability distribution over the types. Using an initial segment of the natural numbers to index the types, a probabilistic selection model for V and N with k types then consists of a probability distribution $\lambda \tau p_\tau$ over the set of integers $\{1, ..., k\}$ $(= T)$, and for each type τ in $\{1, ..., k\}$, a pair of probability distributions $\langle \lambda v p^\tau_v, \lambda n p^\tau_n \rangle$, as described above.

Derivatively, for any type τ we construct a probability distribution on $V \times N$ as a product :

$$p^\tau_{v,n} = p^\tau_v p^\tau_n$$

We constuct a probability distribution on $T \times V \times N$ as a disjoint union of these products:

$$p_{\tau,v,n} = p_\tau p^\tau_v p^\tau_n$$

Such a model can by used to assign probabilities to verb-object pairs. In the probility space just defined, if a verb-noun pair is generated, it is generated in some type, and we obtain the probability for the verb-noun pair by summing over the types:

$$p_{v,n} = \sum_\tau p_{\tau,v,n} = \sum_\tau p_\tau p^\tau_v p^\tau_n$$

In section 5, such probabilities are used to compare two grammatical analyses. Table 1.4 gives

type 1 .204				type 2 .315				type 3 .481			
fall	.344	point	.364	raise	.265	rate	.238	sell	.319	share	.340
rise	.343	cent	.281	reduce	.189	price	.207	buy	.288	stake	.198
gain	.128	pence	.080	cut	.162	cost	.143	hold	.095	stock	.172
drop	.059	yen	.071	lower	.109	debt	.071	acquire	.093	interest	.078
decline	.038	average	.017	increase	.100	tax	.063	purchase	.063	asset	.065
climb	.028	bit	.011	boost	.073	rating	.059	trade	.027	unit	.059
jump	.020	mark	.011	push	.036	dividend	.050	retain	.019	security	.038
plunge	.019	foot	.004	slash	.034	value	.044	dump	.009	bond	.037

Table 1.5: The same model, with verbs and nouns represented only where they are most likely to be generated.

the parameters of a selection model of order three for the 24×24 data.[4] Notice that the noun *stake* is ranked high in both the second and third types.

Estimating a model

Given a selection model and observed verb-object pair $\langle v, n \rangle$, the probability that it is generated in type τ is:

$$\frac{p_{\tau,v,n}}{p_{v,n}}$$

Multiplying by the frequency $f_{v,n}$, we obtain the expected number of occurrences of the event $\langle \tau, v, n \rangle$ given the observed frequency and the model:

$$e_{\tau,v,n} = f_{v,n} \frac{p_{\tau,v,n}}{p_{v,n}}$$

This forms a basis for re-estimating the probability parameters:

$$e_{\tau,v} = \textstyle\sum_n e_{\tau,v,n} \qquad e_{\tau,n} = \sum_v e_{\tau,v,n} \qquad e_\tau = \sum_{v,n} e_{\tau,v,n}$$

$$q^\tau_v = \tfrac{e_{\tau,v}}{e^\tau} \qquad q^\tau_n = \tfrac{e_{\tau,n}}{e^\tau} \qquad q_\tau = \tfrac{e^\tau}{\sum_{n,v} f_{n,v}}$$

The probabilities q, the parameters of the new model, are computed as relative frequencies of expected numbers of events, as determined by the old model. For instance, the probability of the verb *fall* within type 1 would the expected number of occurences of *fall* in that type (given the data and model), divided by the expected number of occurrences of that type of verb-object pair.

The formulas are similar to the Baum-Welch re-estimation formulas for hidden Markov models (Baum 1972).[5] Adapting Baum's result for HMMs, it can be shown that an iterative

[4] Or rather, as one can discover by summing the first column of numbers, the approximate parameters.

[5] Selection models as described here can be viewed as zero-order HMMs, augmented with a second surface vocabulary and associated emission probabilities. That is, given a state (or type), two surface symbols are independently generated. Furthermore, as used here, the types are hidden in the sense that they are given no prior interpretation, and are not observed in the data.

re-estimatiom of parameters produces local improvements in the probability of the observed data given the model, and converges to a local maximum of this probability. Table 1.4 was derived in one hundred iterations starting from a random state, using the frequency data in table 1.2. Table 1.5 is a different way of looking at the same model: each verb or noun is is shown only in the type where it is most likely to be generated, i.e. where $p_\tau p_v^\tau$ (or $p_\tau p_n^\tau$) is maximal. This representation reconstructs the three eight-by-eight boxes of table 1.3.

1.4 Results for a larger data matrix

Storage requirements for the estimation algorithm are modest. There are $|T||V|+|T||N|+|T|$ probability parameters. The re-estimation formulas sum over non-zero frequencies, and the expectations can be computed by summing iteratively over such frequencies. In each step, a frequency $f_{v,n}$ is apportioned among the types according to the ratio $\frac{p_{\tau,v,n}}{p_{v,n}}$, and the portion for a type τ is added to running subtotals of $e_{\tau,v}$ and $e_{\tau,n}$. This procedure requires intermediate storage of the same size as the probability model. Since no random access to the frequencies is required there is no need to represent a $V \times N$ matrix. (This might turn out to be useful. In a larger data matrix based on 60 million words of text, about 3500 verb roots occur with five or more different nouns, and about 7500 noun roots, not counting proper nouns or numbers, occur with five or more verbs. One of these numbers would be multiplied if verbs with complements accompanied by prepositions were included as separate entries.)

The algorithm was implemented in Common Lisp. Tables 1.6, 1.7 and 1.8 give the most probable nouns and verbs in each type of a probabiity model for the 992 × 1027 matrix with thirty-two types, obtained with four hundred iterations. The three blocks of 1.3 are represented here: type 3 is the product type (e.g. *develop software*), changing dimensioned objects (*raise price*) are in type 8, and scalar increments (*rise cent*) in type 26. Type 30 is a related one where the object typically denotes a scalar motion event, such as a decline or an increase. The nouns of types 19 and 29 primarily name people. In the nouns, the split seems to amount roughly to a distinction between powerful, active people (executives, lawyers, officials) and weak or passive ones (workers, clients, and shareholders). Looking at the verbs, the type 19 people are appointed and replaced; the type 29 people are given orders and permissions.[6]

Several types are dominated by a common and semantically empty verb, such as *be* (12), *have* (23), or *make* (31). In these cases, the noun sets are not intuitively coherent, presumably because these verbs impose such weak selectional restrictions. The verb sets are not particularly coherent either, though this is balanced by the fact that most of the verbs have low probabilities. These common verbs also occur in many other types, for instace *be* in top position in type 32, which is an intuitively coherent type.

[6] In verb group 19, a number of items resulting from parsing errors are evident. Presumably, *manage* comes from *managing director* misidentified as a verb phrase.

type 1 .015			
meet	.23606	standard	.06597
set	.15089	record	.05207
be	.04807	goal	.04920
keep	.04205	need	.04453
hit	.03637	requirement	.04152
miss	.03403	level	.03564
exceed	.03322	demand	.03168
reach	.02533	target	.02963
achieve	.01923	high	.02828
fulfill	.01518	stage	.02675
satisfy	.01330	pace	.02155
live	.01241	date	.01860
surpass	.01163	payment	.01819
establish	.01126	expectation	.01764
stress	.00919	point	.01623
eliminate	.00858	limit	.01598

type 2 .021			
end	.10515	year	.11873
trade	.07562	Friday	.09733
say	.06728	week	.08977
work	.06685	month	.08074
begin	.06120	day	.06488
spend	.05976	time	.04695
announce	.02880	today	.04067
close	.02638	Tuesday	.03573
open	.02420	hour	.02881
start	.02219	Monday	.02724
expire	.01716	way	.02458
mark	.01618	Wednesday	.02287
last	.01545	capital	.02116
wait	.01396	talk	.01933
follow	.01147	night	.01526
serve	.01117	war	.01030

type 3 .032			
use	.15863	product	.07639
develop	.09296	system	.04459
produce	.08613	drug	.03406
introduce	.03273	technology	.03110
market	.02933	computer	.02236
sell	.02778	country	.01948
include	.02140	car	.01875
ship	.02109	equipment	.01685
supply	.01770	line	.01666
get	.01602	version	.01500
distribute	.01149	machine	.01410
test	.01145	proceed	.01398
manufacture	.01107	program	.01230
promote	.01035	ton	.01226
install	.00983	software	.01178
build	.00971	model	.01146

type 4 .029			
continue	.06015	operation	.08075
begin	.05647	effort	.07703
launch	.04128	program	.06099
conduct	.03157	campaign	.02961
resume	.02845	production	.02943
expand	.02528	investigation	.02209
start	.02476	service	.02029
be	.02280	process	.01670
say	.02159	work	.01508
halt	.01456	negotiation	.01414
follow	.01335	payment	.01407
ban	.01304	activity	.01383
restrict	.01189	practice	.01290
support	.01187	talk	.01264
oversee	.01148	development	.01256
plan	.01145	use	.01224

type 5 .035			
complete	.08198	sale	.10672
finance	.05656	acquisition	.06132
include	.04693	transaction	.05817
be	.03700	change	.04977
say	.03244	purchase	.04142
approve	.03176	merger	.03092
represent	.02703	order	.02709
announce	.02536	increase	.02646
block	.02272	action	.02278
consider	.02208	takeover	.01939
involve	.01758	project	.01827
propose	.01702	issue	.01792
explore	.01670	offering	.01475
expect	.01570	move	.01278
follow	.01496	use	.01205
discuss	.01474	investment	.01098

type 6 .024			
file	.14736	suit	.11129
follow	.09798	case	.06304
settle	.04932	report	.06176
deny	.03748	decision	.04991
issue	.03688	charge	.04782
hear	.03011	lawsuit	.03991
say	.02876	statement	.03674
dismiss	.02466	appeal	.02439
bring	.02329	complaint	.02390
be	.02301	claim	.02312
confirm	.02157	allegation	.02142
appeal	.02020	ruling	.01916
include	.02019	plan	.01561
review	.01744	action	.01490
reverse	.01527	rumor	.01405
see	.01521	recommendation	.01402

type 7 .023			
play	.09036	money	.20670
spend	.07740	$	.07810
raise	.05364	role	.05575
be	.03725	fund	.05401
get	.03375	cash	.03552
lose	.03372	lot	.02660
cost	.03356	time	.02655
save	.03059	dollar	.02612
use	.02727	part	.02037
put	.02145	million	.01805
lend	.01760	game	.01680
pay	.01726	capital	.01508
invest	.01698	total	.01425
need	.01668	billion	.01380
give	.01620	life	.01291
add	.01526	year	.01152

type 8 .041			
raise	.17768	price	.15348
increase	.09248	rate	.15152
boost	.07753	stake	.04734
reduce	.04806	question	.03490
lower	.04484	rating	.03048
cut	.03509	value	.02881
say	.02377	sale	.02544
offer	.02240	earning	.02379
push	.01979	number	.02363
expect	.01605	level	.02135
keep	.01407	dollar	.01719
disclose	.01198	capital	.01707
double	.01191	revenue	.01661
answer	.01182	size	.01569
bring	.01062	cost	.01369
maintain	.01042	production	.01367

type 9 .023			
reject	.08033	bid	.15795
consider	.06539	proposal	.11781
accept	.05285	offer	.11493
support	.03535	plan	.06008
decline	.03064	comment	.03193
drop	.02975	request	.02591
close	.02525	idea	.02331
submit	.02401	attempt	.01774
be	.02389	claim	.01346
receive	.02352	effort	.01319
approve	.02158	issue	.01301
launch	.02135	strategy	.01216
back	.01941	option	.01175
withdraw	.01771	application	.01093
review	.01711	amendment	.01044
announce	.01688	$	.00988

type 10 .014			
hold	.42625	company	.29762
acquire	.04866	meeting	.09623
call	.04400	stake	.05578
attend	.02986	talk	.04176
schedule	.02436	hearing	.03762
leave	.02433	conference	.03723
say	.02301	position	.02211
tell	.02173	election	.01934
force	.02111	concern	.01578
follow	.01705	discussion	.01367
be	.01290	post	.01207
mine	.01207	stock	.01060
control	.01057	hostage	.01017
seek	.00993	party	.00770
expect	.00783	session	.00729
value	.00756	interest	.00715

type 11 .032			
pay	.21425	cost	.13843
reduce	.14585	debt	.08962
cut	.09985	tax	.05801
increase	.03444	dividend	.03704
cover	.02748	price	.03451
include	.02187	$	.02684
be	.01433	fee	.02370
repay	.01355	expense	.02319
slash	.01313	deficit	.02154
keep	.01256	interest	.01911
limit	.01225	bill	.01868
control	.01161	loan	.01818
avoid	.00944	capital-gains	.01785
trim	.00897	premium	.01715
raise	.00868	amount	.01678
collect	.00847	force	.01579

Table 1.6: Part of a 32 type selection model.

type 12 .093			
be	.86747	part	.03984
become	.02744	way	.02670
find	.01918	time	.02601
see	.00639	president	.02546
give	.00600	company	.01500
include	.00383	reason	.01314
get	.00316	lot	.01245
identify	.00295	problem	.01181
remain	.00257	chairman	.01181
cite	.00242	issue	.01158
represent	.00222	case	.01024
seem	.00217	unit	.00933
prove	.00217	sign	.00926
allow	.00201	thing	.00897
want	.00200	target	.00818
mark	.00197	question	.00776

type 13 .025			
do	.20440	business	.15112
run	.10726	job	.09334
be	.09109	thing	.05915
create	.06320	company	.03691
form	.05178	work	.03633
leave	.03496	venture	.02406
get	.03488	lot	.01903
find	.01891	government	.01850
start	.01579	fund	.01547
see	.01489	program	.01358
eliminate	.01072	deal	.01143
keep	.00987	ad	.01139
lose	.00979	risk	.00982
expect	.00978	year	.00949
enter	.00822	system	.00826
establish	.00798	room	.00784

type 14 .030			
show	.10237	interest	.07597
reflect	.05022	growth	.05279
express	.03955	value	.03758
see	.03130	economy	.03269
improve	.02787	concern	.03047
slow	.01795	demand	.03012
say	.01794	sign	.02552
enhance	.01611	performance	.02153
continue	.01535	return	.02080
pursue	.01512	confidence	.02062
grow	.01505	ability	.02002
fuel	.01503	strength	.01637
indicate	.01460	inflation	.01435
cite	.01430	benefit	.01353
represent	.01363	support	.01348
expect	.01304	improvement	.01302

type 15 .022			
reach	.14872	agreement	.23792
sign	.11051	plan	.15452
announce	.08203	bill	.08091
approve	.06145	contract	.04619
pass	.04008	legislation	.03891
be	.03342	letter	.02473
negotiate	.02940	accord	.02395
say	.02620	settlement	.02249
introduce	.01946	deal	.01928
terminate	.01562	measure	.01622
spend	.01310	pact	.01546
veto	.01293	package	.01450
adopt	.01238	level	.01253
propose	.01171	resolution	.01110
forge	.01142	program	.01028
discuss	.01009	decision	.00951

type 16 .016			
send	.13358	letter	.06276
carry	.08868	dividend	.05273
receive	.06249	warrant	.04434
write	.05310	message	.03976
declare	.04667	book	.02798
get	.04335	note	.02597
publish	.03554	signal	.02243
issue	.02821	stock	.02046
read	.02800	article	.02037
deliver	.02297	information	.01688
include	.01517	sentence	.01583
serve	.01473	ad	.01531
subordinate	.01435	price	.01447
see	.01412	news	.01350
suspend	.01406	report	.01294
regard	.01123	copy	.01098

type 17 .020			
increase	.08773	market	.25364
expand	.07881	pressure	.06255
put	.06053	business	.03365
enter	.05256	board	.02548
be	.03619	capacity	.02433
grow	.02317	industry	.02232
tap	.01821	membership	.02207
dominate	.01791	line	.01559
keep	.01595	economy	.01522
open	.01550	area	.01442
serve	.01435	base	.01225
hit	.01415	end	.01221
putt	.01408	competition	.01185
affect	.01397	country	.01128
bring	.01250	sale	.01123
help	.01201	head	.01119

type 18 .021			
violate	.07151	law	.08733
impose	.04621	right	.08309
adopt	.04096	rule	.07114
ease	.03129	policy	.05715
exercise	.03047	provision	.04352
include	.02975	restriction	.04050
tighten	.02341	credit	.03410
propose	.02050	ban	.03290
enforce	.01916	regulation	.02586
use	.01698	security	.02519
change	.01655	control	.02277
break	.01621	option	.02191
extend	.01587	standard	.01958
lift	.01441	requirement	.01490
oppose	.01376	tax	.01227
remove	.01323	duty	.01227

type 19 .031			
say	.26679	director	.08356
become	.11146	president	.06178
manage	.07970	official	.05558
be	.06654	analyst	.04518
remain	.05788	chairman	.04354
tell	.03499	executive	.04324
include	.03194	manager	.03607
name	.02965	partner	.02709
hire	.02161	member	.02248
market	.01336	trader	.01837
act	.00933	board	.01599
elect	.00929	firm	.01408
oust	.00924	lawyer	.01362
ask	.00854	company	.01347
appoint	.00786	consultant	.01221
replace	.00770	banker	.01196

type 20 .015			
assume	.05587	position	.12880
fill	.04523	account	.06505
retain	.04154	control	.04908
return	.03565	power	.04743
be	.02827	image	.04092
share	.02387	responsibility	.03439
strengthen	.02213	post	.03189
maintain	.02148	call	.03184
seize	.02047	view	.01745
bear	.01857	home	.01714
change	.01815	ownership	.01703
improve	.01648	job	.01595
use	.01636	order	.01501
shift	.01626	name	.01296
place	.01532	title	.01296
bolster	.01465	seat	.01233

type 21 .016			
join	.12423	firm	.13008
lead	.09514	mortgage	.06931
cap	.07303	group	.05734
bank	.07174	company	.05733
consult	.06777	attention	.05347
head	.05825	concern	.03268
draw	.04459	force	.02934
indicate	.03417	coupon	.02364
engineer	.02217	list	.02076
trade	.01944	indicator	.01819
attract	.01666	unit	.01787
form	.01528	office	.01772
stage	.01513	board	.01306
focus	.01419	operation	.01214
turn	.01066	team	.01078
create	.01065	way	.01048

Table 1.7: Another part.

type 22 .025			
operate	.23595	officer	.08099
build	.12124	plant	.08029
open	.05096	profit	.05795
close	.03310	system	.04020
manufacture	.02973	store	.03467
keep	.02182	facility	.03319
run	.01790	operation	.02793
turn	.01747	office	.02522
establish	.01689	center	.02376
move	.01551	car	.01748
be	.01425	company	.01672
fly	.01396	home	.01634
expand	.01335	door	.01471
say	.01300	mile	.01144
maintain	.01149	building	.01131
lease	.01147	eye	.01127

type 23 .053			
have	.88786	loss	.03589
be	.02008	share	.03026
say	.00934	effect	.02858
get	.00734	impact	.02606
lose	.00644	sale	.02217
see	.00606	interest	.02082
limit	.00577	income	.01999
lack	.00458	problem	.01842
increase	.00432	right	.01549
cite	.00375	plan	.01473
cast	.00366	time	.01334
add	.00228	chance	.01282
exceed	.00220	trouble	.01257
feel	.00208	comment	.01108
consider	.00198	value	.01099
mean	.00194	asset	.01065

type 24 .058			
sell	.24706	share	.17737
buy	.21141	stock	.09721
acquire	.06323	stake	.06503
purchase	.04274	company	.03334
own	.03560	interest	.03095
include	.03528	asset	.03044
total	.02347	unit	.02787
hold	.02227	bond	.02452
be	.01894	business	.02242
issue	.01459	security	.01817
prefer	.01360	operation	.01475
trade	.01205	dollar	.01390
retain	.00957	ton	.01164
offer	.00840	issue	.01091
receive	.00702	product	.00996
increase	.00684	car	.00955

type 25 .031			
receive	.15526	approval	.09046
get	.14544	contract	.08712
win	.11622	control	.04008
seek	.11268	order	.02775
gain	.07195	support	.02298
obtain	.03624	share	.01816
give	.02811	damage	.01665
require	.02492	license	.01496
lose	.02422	access	.01461
need	.02124	help	.01358
loose	.01948	benefit	.01258
grant	.01426	vote	.01237
demand	.01242	right	.01227
secure	.01030	loan	.01140
award	.00975	attention	.01126
await	.00788	boost	.01075

type 26 .054			
rise	.22903	%	.74167
yield	.13168	point	.07754
fall	.12219	cent	.05860
be	.08785	yen	.01537
own	.03445	pence	.01498
drop	.02992	price	.00969
jump	.02585	rate	.00935
grow	.02474	year	.00776
increase	.02322	average	.00451
climb	.02222	cost	.00328
decline	.02091	bit	.00276
gain	.02001	demand	.00253
hold	.01557	ton	.00250
buy	.01167	fee	.00244
acquire	.01089	inflation	.00244
soar	.00896	range	.00231

type 27 .029			
provide	.22624	service	.05067
offer	.11291	way	.04967
change	.08438	information	.04529
give	.06825	detail	.04282
find	.04567	hand	.04176
disclose	.03263	name	.02648
get	.03251	data	.02517
be	.01960	benefit	.01551
seek	.01937	term	.01436
discuss	.01320	incentive	.01346
use	.01279	figure	.01076
clear	.01147	evidence	.01069
include	.01115	$	.01045
obtain	.01086	loan	.01006
need	.01085	reason	.00998
release	.00901	record	.00987

type 28 .025			
take	.71341	yesterday	.08897
trade	.21074	place	.06415
handle	.00380	advantage	.05361
offer	.00343	step	.04657
call	.00334	action	.03615
include	.00324	effect	.03290
see	.00264	volume	.03119
find	.00205	charge	.02457
offset	.00204	control	.02322
represent	.00179	position	.01873
enjoy	.00148	year	.01862
remain	.00146	profit	.01806
accept	.00121	look	.01770
give	.00120	time	.01435
overcome	.00117	risk	.01336
note	.00112	part	.01320

type 29 .052			
allow	.05595	company	.07287
give	.04996	people	.05791
tell	.04769	investor	.04724
help	.04348	government	.02802
ask	.03068	customer	.02600
require	.02369	employee	.02555
say	.02168	worker	.02435
attract	.02134	client	.01987
force	.02101	bank	.01928
represent	.02067	shareholder	.01402
protect	.01823	agency	.01363
leave	.01805	consumer	.01271
urge	.01768	group	.01270
keep	.01565	state	.01254
include	.01482	reporter	.01234
encourage	.01475	court	.01153

type 30 .037			
report	.19567	loss	.17676
post	.12245	gain	.10309
say	.07210	earning	.09973
expect	.05355	profit	.08199
include	.04197	income	.05284
operate	.02780	sale	.04715
show	.02579	decline	.04575
attribute	.02379	increase	.04519
follow	.02049	result	.04247
have	.01666	rise	.02973
generate	.01453	drop	.02688
produce	.01249	revenue	.02411
be	.01199	event	.01580
see	.01023	net	.01198
reflect	.01005	return	.01168
estimate	.00984	charge	.00803

type 31 .029			
make	.83180	decision	.05421
say	.01986	money	.03619
call	.00735	payment	.03542
see	.00696	sense	.03247
expect	.00582	bid	.02889
accept	.00552	offer	.02289
welcome	.00513	product	.02105
use	.00451	change	.02097
avoid	.00400	loan	.01927
leave	.00360	move	.01693
keep	.00357	profit	.01682
affect	.00315	investment	.01484
guarantee	.00299	difference	.01479
cancel	.00278	effort	.01280
defer	.00277	statement	.01278
reflect	.00250	$	.01184

type 32 .029			
be	.11257	problem	.14799
face	.08331	issue	.02183
cause	.04889	pressure	.02099
resolve	.02475	damage	.02080
solve	.02389	recession	.01835
avoid	.02373	challenge	.01664
address	.02302	situation	.01663
fight	.02070	effect	.01617
create	.02057	competition	.01527
pose	.01971	threat	.01394
reflect	.01856	risk	.01369
see	.01673	concern	.01343
prevent	.01505	question	.01327
ease	.01416	crisis	.01323
suffer	.01239	shortage	.01284
cite	.01236	crime	.01239

Table 1.8: Still another part.

1.5 Application to parsing

A casual examination of clusters can at most suggest that the right sort of thing is going on; the point is to use such representations to do something that we want to do for an independent reason. In the introduction, I said that resolving many parsing ambiguities comes down to evaluating selectional compatibility between two lexical items. Given a probabilistic selection model, we can assign a probability to any verb-object pair drawn from the lexicon of 992 verbs and 1027 nouns. In order to evaluate parses, we need to include other grammatical relations. In some cases, such as subjects, this is fairly straightforward. In others, such as second complements of verbs, getting access to frequency counts is not straightforward, since identifying second complements — such as prepositional phrases — requires resolving attachment ambiguities. This can not be done systematically without the kind of lexical information we are trying to induce. Presumably, a procedure learning selectional restrictions for second complements would have to initially consider several attachments, and iteratively learn the lexical information required for disambiguation. This method is applied to simpler data (paying attention just to prepositions and not the heads of their objects) in Hindle and Rooth (1993).

To investigate the possibility of disambiguating syntactic ambiguities with selectional information, I considered the past participle vs. tensed verb ambiguity of *sold* in positions immediately following a noun phrase. In the first example below, *sold* is a tensed verb, and the preceding noun *administration* is the head of its subject, describing the agent. In the second example, *sold* is a participial post-modifier, and the preceding noun phase denotes the sold object.

(11) a. 10604815 evidence that the Reagan administration sold arms to Iran
b. 3001228 represented payments for arms sold to Nicaraguan insurgents

The situation is actually more complex, since *sold* occurs frequently in the middle construction, with a syntactic subject filling the semantic role of a sold object, rather than the seller:

(12) 2979922 The franchise sold in 1979 for $11 million

To sidestep this problem, I defined the problem to be solved as one of identifying the semantic relation (seller or sold object) of the noun phrase, rather than identifying a grammatical relation or part of speech. In other words, the middle constructions are grouped with the postmodifiers. By hand, I classified the first relevant occurrence of each noun-*sold* pair in the full Wall Street Journal corpus from Liberman (1992). Of the 1027 nouns in the model, 220 were represented in this configuration, of which 87 were past tense verbs, 118 were participial postmodifiers, and 15 middle constructions. An augmented selection model was obtained (in a somewhat ad hoc way) by adding two additional verbs sold/VBN and sold/VBD to the verb set. Training material consisted similar but independent data.[7] The model was used to classify the 220 test examples by means of the ratio:

offering	.13354	metal	.14244	apartment	.14495	stock	.14550
franchise	.15312	membership	.16785	future	.19601	seat	.27562
price	.43852	hundred	.44699	conference	.49000		
other	.57470	thing	.65876	target	.99409	run	.99997

Table 1.9: Disambiguation scores for middle constructions. Items above the line are correctly classified.

$$\frac{p_{\text{sold/VBD},n}}{p_{\text{sold/VBD},n} + p_{\text{sold/VBN},n}}$$

Pairs with a score greater than 0.5 were assigned to the seller role, and those with a lower score to the sold object role. Of the 118 post-modifier pairs, 110 were correctly classified into the sold object role. Of the 87 ordinary subject-verb items, 69 were correctly classified into the seller role. Of the 15 instances of the middle construction, 11 were correctly classified into the sold object role. This gives an rate of correct classification of 86.4%. The disambiguation scores are listed in tables 1.10, 1.11, and 1.9. Appositely, the least ambiguous instance of the seller role is the noun *seller*. The least ambiguous example of the sold object role is *output*. Many of the problematic nouns — those with scores below 0.5 in table 1.11 — name institutions which can be agents, but can also be bought and sold, for instance *company*, *airline*, and *store*. Since companies are actually described in the Wall Street Journal both as being sold and as selling things, we would not expect a selectional approach to be uniformly successful in this case. However, we want to resolve clear cases correctly — artefacts, materials, and financial instruments are unambiguos sold objects, and people are unambiguous sellers. With few exceptions, such clear cases are resolved correctly.

1.6 Matrix formulation

The definition of $p_{v,n}$ from section 3 can be written as a matrix product. Let L be the $V \times k$ matrix representing the verb probabilities, $L_{i,\tau} = p_i^\tau$, let R be the $N \times k$ matrix representing the noun probabilities, $R_{j,\tau} = p_j^\tau$, and let D be a diagonal matrix representing the type probabilities, $D_{\tau,\tau} = p_\tau$. Then a derived probability distribution on $V \times N$ is given by a matrix product:

$$LDR^T \qquad [L_{v/t} D_{t/t} [R^T_{n/t}]_{t/n}]_{v/n} \tag{1.1}$$

In the version on right, the subscripts give matrix dimensions in categorial notation, v being the verb cardinality, t the type cardinality, and n the noun cardinality; R^T is the transpose of the matrix R, $[R^T]_{i,j} = R_{j,i}$.

[7] In writing this section, I found that the training data and my record of how they were constructed had been lost. Therefore, the evaluation will be redone, and final results may differ from those described here.

output	.00000	volume	.00001	suit	.01225	gasoline	.04458
bill	.04526	missile	.05726	movie	.07933	system	.08979
technology	.09182	package	.09475	film	.09783	advertising	.11294
material	.11427	shoe	.11488	oil	.12449	product	.12479
chip	.12479	plant	.12868	device	.13161	merchandise	.13183
good	.13240	car	.13363	machine	.13475	operation	.13514
barrel	.13680	building	.13923	loan	.14066	document	.14141
gold	.14160	copy	.14185	debt	.14402	truck	.14454
debenture	.14468	bond	.14542	brand	.14633	share	.14641
gas	.14673	food	.14708	coverage	.14758	certificate	.14839
inventory	.14886	stake	.14924	asset	.14957	acre	.15149
phone	.15157	property	.15334	note	.15414	version	.15509
warrant	.15515	ticket	.15528	block	.15751	card	.15806
pound	.16148	business	.16485	home	.16962	contract	.17100
computer	.17181	steel	.17360	show	.17449	amount	.17630
dollar	.18039	program	.18111	aircraft	.18204	insurance	.18227
engine	.18332	book	.18543	site	.19478	line	.19487
land	.19565	drug	.19588	security	.19686	weapon	.19701
test	.19931	tape	.20481	item	.21203	house	.21365
rest	.21706	plane	.21825	station	.21854	piece	.22123
paper	.22272	mark	.22307	import	.22753	arm	.23139
model	.23296	magazine	.23470	call	.24358	billion	.24488
service	.25488	issue	.25688	work	.25714	policy	.26237
energy	.26286	supply	.26545	position	.29452	vehicle	.29797
thrift	.30081	collection	.30386	list	.33147	ad	.33409
million	.33923	game	.34198	one	.34246	shipment	.38118
art	.38336	article	.39738	newspaper	.39756	type	.41898
set	.43282	transaction	.45646				
hospital	.55241	kind	.68506	player	.69998	export	.71760
switch	.74617	agent	.75588	animal	.84308	premium	.84474

Table 1.10: Disambiguation scores for NP/postmodifier combinations. Items above the line are correctly classified.

trade	.00001	effort	.08683	plan	.11405	shop	.14468
estate	.16620	division	.17522	store	.17609	unit	.23540
subsidiary	.26404	account	.27786	network	.32662	fund	.32709
company	.36211	trust	.39613	partnership	.40965	concern	.43930
airline	.44054	institution	.46481				
giant	.50438	team	.54480	parent	.59502	organization	.63279
operator	.63752	utility	.63796	insurer	.64974	bank	.66487
industry	.66728	couple	.67811	manager	.68160	maker	.70603
country	.71725	firm	.72093	manufacturer	.72645	shareholder	.74399
foreigner	.75349	room	.75649	lender	.75983	group	.77943
corporation	.78073	husband	.80379	publisher	.80626	developer	.81175
nation	.83127	stockholder	.83832	state	.87692	family	.88925
school	.89850	broker	.90178	individual	.90353	executive	.90404
producer	.90640	agency	.91013	partner	.93265	customer	.94037
wife	.94645	chairman	.94838	holder	.95723	board	.95960
man	.96147	department	.96479	father	.96875	dealer	.97063
people	.97064	member	.97166	investor	.97463	brother	.97580
employee	.97603	plaintiff	.97753	world	.97878	government	.98305
defendant	.98917	farmer	.99021	friend	.99417	lawyer	.99551
trader	.99737	official	.99850	client	.99883	owner	1.00000
administration	1.00000	analyst	1.00000	buyer	1.00000	director	1.00000
foundation	1.00000	officer	1.00000	participant	1.00000	president	1.00000
seller	1.00000						

Table 1.11: Disambiguation scores for agent/verb combinations. Items below the line are correctly classified.

Proposition. Suppose L,D, and R are probability matrices as described above, that is:

$$\forall\tau\left[\textstyle\sum_i L_{i,\tau}=1\right]\quad\forall\tau\left[\textstyle\sum_j R_{j,\tau}=1\right]\quad\textstyle\sum_\tau D_{\tau,\tau}=1$$

Then $\sum_{i,j}[LDR^T]_{i,j}=1$. In the verification below, $M_{i\cdot}$ and $M_{\cdot j}$ denote the ith row and jth column of a matrix M, respectively.

$$\begin{array}{rcll}
[DR^T]_{\tau,j} & = & D_{\tau\cdot}\cdot[R^T]_{\cdot j} & \\
 & = & D_{\tau\cdot}\cdot R_{j\cdot} & \\
 & = & D_{\tau,\tau}R_{j,\tau} & \text{(since } D \text{ is diagonal)} \\
[LDR^T]_{i,j} & = & L_{i\cdot}\cdot[DR^T]_{\cdot j} & \\
 & = & \sum_\tau L_{i,\tau}[DR^T]_{\tau,j} & \\
 & = & \sum_\tau L_{i,\tau}D_{\tau,\tau}R_{j,\tau} & \text{(previous equality)} \\
\sum_{i,j}[LDR^T]_{i,j} & = & \sum_{i,j}\sum_\tau L_{i,\tau}D_{\tau,\tau}R_{j,\tau} & \text{(previous equality)} \\
 & = & \sum_\tau D_{\tau,\tau}\left(\sum_i L_{i,\tau}\left(\sum_j R_{j,\tau}\right)\right) & \\
 & = & \sum_\tau D_{\tau,\tau}\left(\sum_i L_{i,\tau}\right) & \text{(second assumption)} \\
 & = & \sum_\tau D_{\tau,\tau} & \text{(first assumption)} \\
 & = & 1 & \text{(third assumption)}
\end{array}$$

So, we are justified in describing the matrix product as a probability distribution on $V \times N$. This representation is reminiscent of singular value decompositon of matrices (SVD).: in fact such a decompositon takes exactly the form (1.1). However, the constraints on L and R are different: in SVD, they are orthonormal, meaning that any two distinct rows a null dot product, and the dot product on any row with itself is 1. These conditions are different from those imposed by the interpretation as probability matrices. One symptom of this is that a SVD approximation of a frequency matrix may contain negative entries.

A further difference has to do with the relation beween the product matrix and the original data. In SVD, the product provides the best least squares fit to the frequency matrix, of a given rank. What is being optimized in the probability model is most easily understood as the probability of the observed sequence of verb-noun pairs. This can be written:

$$\prod_{v,n} p_{v,n}^{f_{v,n}}$$

In entropy terms, we seek to minimize

$$\sum_{v,n} f_{v,n}\left(-\log_2 p_{v,n}\right)$$

This is quite different from the least-squares criterion.

Valence Induction with a Head-Lexicalized PCFG

– Gold – *

Glenn Carroll and Mats Rooth

Abstract

In current theory, a key portion of the lexicon is a specification of *subcategorization frames* or *valences* for open-class words. A computational lexicon of useful size requires a vocabulary of several thousand words, each of which may have dozens or hundreds of frames, depending on the desired granularity. Moreover, the usage of such frames is subject to constant innovation, so that the task of constructing the lexicon for a living language is never done.

This paper addresses the problem of acquiring valences by means of a learning technique based on head-lexicalized probabilistic context free grammars and the expectation-maximization (EM) algorithm. Given a hand-written grammar and a text corpus, a rather loose vaiation of the EM algorithm is employed to estimate the distribution of frames given head words. We show that the lexicon acquired by this technique is comparable or better than other automatically acquired lexica, and in some ways superior to a published, manually-acquired reference.

* This paper is a longer (and earlier) version of Carroll and Rooth (1998). Valence induction with a head-lexicalized PCFG. In *Proceedings of EMNLP-3*, Granada. It differs primarily in the inclusion of an explanation of smoothing.

2.1 Introduction

In contemporary linguistic and computational linguistic theories, the lexicon for a natural language includes a specification of *subcategorization frames* or *valences* for open-class words such as verbs and nouns. For instance, the lexicon for English specifies that the verb *order* may (among other possibilities) combine to form a verb phrase with:

1. a noun phrase (*order · a major pullback, order · a pot of tea*)
2. a noun phrase followed by a to-infinitive, with either an object control or raising-to-object interpretation (*order · the FDA · to speed drug approval, order · the Bay of Pigs invasion · to go forward*)
3. a noun phrase followed by a passive verb phrase (*order · the meeting · delayed, order · the documents · suppressed*).

Depending on theoretical framework, such frames are specified more or less directly (as in categorial grammar) or factored into components which entail realized surface frames indirectly (e.g.Pollard and Sag (1987), Stowell (1981)).

Whatever the details of representation, the need to describe subcategorization frames for whole languages constitutes a challenge of scale. Written languages have vocabularies of tens of thousands of words, and there are tens of possible frames, or many more if lexically determined particles and prepositions are counted as part of a frame. Furthermore, lexical properties such as subcategorization are subject to constant innovation, so that the task of constructing the lexicon for a living language is never done. Finally, there are quite a lot of languages for which comprehensive computational linguistic lexica are scientifically and practically important.

This paper addresses this problem by means of a learning technique based on probabilistic lexicalized context free grammars and the expectation-maximization (EM) algorithm. Given a hand-written grammar and a text corpus, expected frequencies of a head word accompanied by a frame are calculated, and such frequencies are used to compute probability parameters characterizing subcategorization. Since the calculation of expectations uses a probabilistic weighting of alternative analyses, the procedure can be iterated, resulting in improved models of subcategorization.

A further problem addressed here concerns lexically driven conditioning of lexical choice; we define a model in which each word is probabilistically conditioned on another word and a syntactic environment. For instance, in *order a major pullback*, the choice of the object head *pullback* is conditioned on the verb *order*, and the adjective *major* and determiner *a* are conditioned on the nominal head *pullback*.

Section 2 describes the grammar formalism and a specific grammar of English. In section 3 is concerned with the probability model; we define certain probability parameters and explain

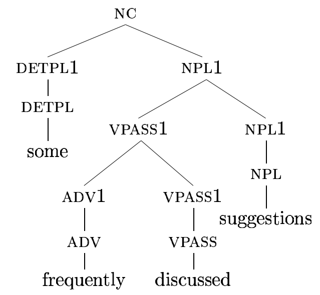


Figure 2.1: A noun chunk

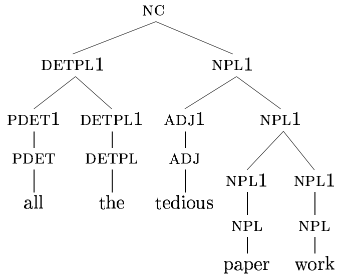


Figure 2.2: Another noun chunk

how they induce a probabilistic weighting of trees. Section 4 explains parameter estimation via the EM algorithm, a procedure which allows probability parameters to be determined given the grammar and a text corpus. Section 5 describes an experiment in extracting a probabilistic subcategorization lexicon from the British National Corpus (BNC); results are evaluated both informally and formally in section 6. In the final section we sketch directions for future work.

2.2 A grammar and formalism

The core of the grammar used in our experiment is a conventional $\bar{X}$ grammar of phrases including noun phrases, prepositional phrases, and verbal clusters. Some representative nominal chunks are given in figures 2.1 and 2.2.

The symbol NC is read "noun chunk"; similarly we work with adjective chunks (ADJC), finite verbal chunks (VFC), prepositional chunks (PC), and so forth. Our use of this concept is inspired by Abney (Abney 1991; Abney 1995). The reason for distinguishing chunks from phrases such as NP, AP, VP and so forth is that complements and trailing adjuncts are not included in chunks. For instance a verbal chunk generally ends with the head lexical verb, so that complements following the verb are excluded.

Within NC, we see a familiar $\bar{X}$ structure (e.g. Jackendoff (1977)). Modification takes place by adjunction of a single bar category to a single bar category. In the first tree, [$_{\text{ADV1}}$ frequently] is a modifier of [$_{\text{VPASS1}}$ discussed], [$_{\text{VPASS1}}$ frequently discussed] is a modifier of [$_{\text{NSG1}}$ suggestion], and [$_{\text{PDET1}}$ all] is a modifier of [$_{\text{DET1}}$ the]. Embedding of chunk categories in other chunk categories is not excluded — for instance an NC can embed another NC as a genitive.

Verbal categories are also based on an $\bar{X}$ scheme. Some examples are given in figures 2.3 and 2.4. Verbal category labels are based on a feature decomposition in four dimensions:

1. Bar level.

2. Inflectional form. F, TO, N, and G correspond fairly closely to the VFORM feature of Gazdar et al. (1985). F indicates a finite form, N a past participle, and G a present participle. TO indicates the verb or infinitive marker *to*.

3. Auxiliary verbs have distinguished categories: M indicates modal, H indicates *have*, and B indicates *be*.

4. Main verbs are classified as passive versus non-passive.

Thus for instance VFC is a finite non-passive verbal chunk, and VFPC is a finite passive verbal chunk; VF2 is the two-bar projection of a finite non-passive main verb, and VHF2 is the two-bar projection of the auxiliary verb *have*; VGC is a present participle chunk, and VNC is a past participle chunk.

As described above, the categories are often interpretable in terms of a feature decomposition. However, they are treated as atomic in the formalism. Thus NSG1 and NPL1 are distinct atomic symbols. We depart from a standard context-free formalism in that heads are marked on the right hand sides of rules, using a prime (′). Here are some of the rules used in licensing the noun chunks in the figures.

NSG1 → VPASS1 NSG1′
NSG2 → NSG′
DETSG1 → PDET1 DETSG1′

As explained later, the head marking is used in the lexicalization of the probabilistic grammar.

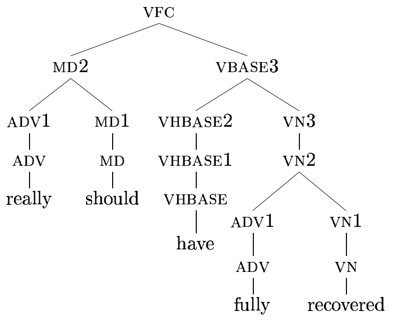


Figure 2.3: A finite verb chunk

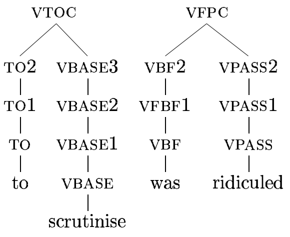


Figure 2.4: More verb chunks

Complementation rules

The grammar includes complementation rules for verbs, nouns, and adjectives, and prepositions. Complements are attached at a level above the chunk; for instance, the category VFP is expanded as a finite verb chunk VFC and a sequence of complements. The following are examples of rules expanding VFP.

VFP → VFC′	is rustling
VFP → VFC′ NP	rubbed · his spine
VFP → VFC′ AP	has remained · very sick
VFP → VFC′ PP	was musing · on the problem
VFP → VFC′ VTOP	may have decided · to depart
VFP → VFC′ NP PP	put · the book · on the desk
VFP → VFC′ PARTP	runs · away
VFP → VFC′ NP VTOP	convince · her · to stay
VFP → VFC′ PP PP	talked · to her · about it

We call categories such as NP, P, VFP and VTOP phrasal categories; such categories end in the letter P. Thus the phrasal category VFP expands as a finite verb chunk VFP plus its complements.

Examples of phrases covered by the rules are given in the last column. For instance, the third rule expands VFP as VFC AP; *has become* is a VFP and *very sick* is an AP.

There are similar complementation rules for other verbal categories, and for noun phrases, adjective phases and prepositional phrases. Figure 2.5 gives a tree involving one verbal complement, one nominal one, and one prepositional one.[8]

In addition to phrasal categories which consist of a corresponding head chunk category and its complements, there are phrasal categories which are not headed by chunk categories. These are phrases with no complements such as adverb phrases; a special case of these are those phrases which always consist of a single word, such as punctuation.

The state grammar

The third and least standard part of the grammar is a large set of *state* or *n-gram* rules which put together a parse without constructing a standard clause-level analysis. Phrasal categories are strung together with context-free rules modeling a finite state machine, with states expressed as categories consisting of an ordered pair of phrasal categories. This results in right-branching structures, as illustrated figure 2.6. Note that the entire tree in figure 2.5

[8] Here there is a disjuncture between the grammar being described in this section and the one used for the experiment in section 5. In that grammar, noun complements were attached at the N1 level, so that NC had the status of the phrasal category NP described here. This lack of uniformity in the treatment of complementation was eliminated in the next generation of the grammar.

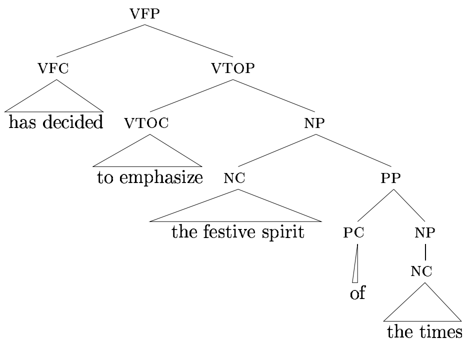


Figure 2.5: Complementation

could be substituted for the finite verb phrase VFP in figure 2.6.

The backbone of the tree is the chain of categories NP:VFP VFP:COMC COMC:ADVC ADVC:PERC. These categories represent the conditioning of one phrasal category on the previous two such categories. Given the state NP:VFP a succeeding category COMP is chosen, and the machine enters the state VFP:COMP. All category sequences are possible: for all chunk and phrase categories X, Y, Z, the grammar includes a rule:

$$\text{X:Y} \quad \rightarrow \quad \text{Y}' \quad \text{Y:Z}$$

Since the possibilities are entirely open, the n-gram rules provide no constraint in the non-probabilistic form of the grammar. In a probabilistic version, however, such constraints are imposed, providing a useful and robust (though linguistically unrealistic) model of higher-level sentence structure.

To complete this section, we define headed context-free grammars in the sense employed here, together with several notions used in the next section.

Definition. A headed context free grammar is a tuple $\langle N, T, W, \mathcal{L}, \mathcal{R}, s \rangle$, where

1. N and T are disjoint sets, interpreted as the non-terminal and terminal categories respectively.
2. W is a set, interpreted as the set of words.
3. $\mathcal{L}$ is a relation between T and W, characterizing the possible realizations of a given terminal category as a word. $\mathcal{L}$ and N are required to be disjoint.

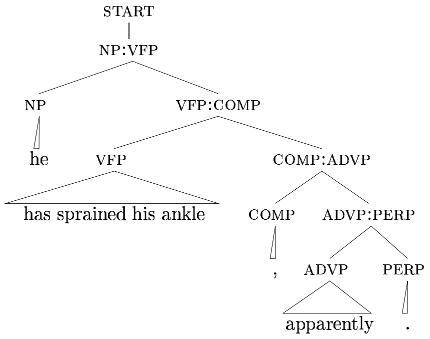


Figure 2.6: Structure above the phrase level

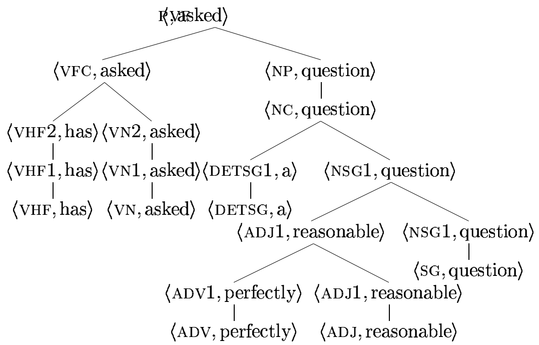


Figure 2.7: Lexicalization of categories by projection of heads of words

4. $\mathcal{R}$ is a finite subset of $N \times N^* \times (N \cup T) \times N^*$, the set of headed productions.

5. $s \in N$, with the interpretation of a start symbol.

We typically use $\bar{n}$ as a variable for mother categories, n for the head daughter category, and α and β for the category sequences flanking the head on the right hand side, so that $\langle \bar{n}, \alpha, n, \beta \rangle$ represents a rule. x is used for non-head categories. Note that the sequences α and β are required to be elements of N^* rather than $(N \cup T)^*$, so that terminal categories are not permitted to occur as non-heads on the right hand side of productions.

To illustrate the representation of rules, the production

VFP → VFC′ NP PP

is encoded as a four-tuple the first element of which is the left hand side VFP; the second element of which is the empty sequence, because there are no categories preceding the head; the third element of which is the head category VFC; and the fourth element of which is the sequence NP,PP or $\langle$VFP, {}, VFC, $\langle$NP,PP$\rangle\rangle$.

A category n in $N \cup T$ is a *possible immediate head* of a category $\bar{n}$ in N if there is some rule of the form $\langle \bar{n}, \alpha, n, \beta \rangle$. The set of *lexicalized nonterminals* $\widehat{N} \subseteq N \times W$ is the composition of the transitive closure of the possible-immediate-head relation with $\mathcal{L}$. We have $\langle y, w \rangle \epsilon \mathcal{N}$ exactly if the word w can be the lexical head of the nonterminal category y.

2.3 Lexicalization and the probability model

Head marking is used to project lexical items up a chain of heads. In the transitive verb phrase in figure 2.7, *question* is projected to the NP level, and *asked* is projected to the VFP level.

In this tree, the non-terminal nodes are lexicalized non-terminals, while the terminal nodes are members of $\mathcal{L}$.

The point of projecting head words is to make information which probabilistically conditions rules and lexical choices available at the relevant level. At the top level in this example, the head *asked* is used to condition the choice of the phrase structure rule

VFP → VFC′ NP

as well as the choice of *question*, the head of the object. We now define events which characterize such choices of rules and of lexical heads.

Definition. Given a grammar $G = \langle N, T, W, \mathcal{L}, \mathcal{R}, s \rangle$ with lexicalized non-terminals $\mathcal{N}$, the set of rule events $ER(G)$ is the set of tuples $\langle \bar{w}, \bar{n}, \alpha, n, \beta \rangle$ such that $\langle \bar{n}, \bar{w} \rangle$ is an element of $\mathcal{N}$ and $\langle \bar{n}, \alpha, n, \beta \rangle$ is an element of $\mathcal{R}$. The set of lexical choice events $EL(G)$ is the set of tuples $\langle \bar{w}, \bar{n}, x, w \rangle$ such that:

1. $\langle \bar{n}, \bar{w} \rangle$ is an element of $\mathcal{N}$;

2. in some rule of the form $\langle \bar{n}, \alpha, n, \beta \rangle$, x is an element of one or both of the category sequences α and β;

3. $\langle x, w \rangle$ is an element of $\mathcal{N}$.

By virtue of the length of tuples, $ER(G)$ and $EL(G)$ are disjoint, and the union $E(G)$ can be formed without confusing lexical with rule events. We now define probabilistic head-lexicalized grammars.

Definition. Let G be a headed context free grammar. A head-lexicalized probabilistic context free grammar with signature G is a function p with domain $E(G)$ and range $[0, 1]$ satisfying the conditions below.

1. Fixing any lexicalized non-terminal $\langle \bar{n}, \bar{w} \rangle$,
$$\sum_{\alpha,n,\beta} p_{\bar{w},\bar{n},\alpha,n,\beta} = 1$$

2. Fixing any lexicalized non-terminal $\langle \bar{n}, \bar{w} \rangle$ and possible non-head daughter x,
$$\sum_{x,w} p_{\bar{w},\bar{n},x,w} = 1$$

Here we are writing the value of the function p on a rule event as $p_{\bar{w},\bar{n},\alpha,n,\beta}$, and on a lexical event as $p_{\bar{w},\bar{n},x,w}$.

The definition characterizes p as a representation of a family of probability measures; representation by a function defined on the union of the sample spaces for the measures is possible because the sample spaces were constructed to be disjoint. The function p gives probabilities for elementary events; probabilities for non-elementary events are defined by summation.

To assign probability weights to trees, we define a tree-licensing and labeling interpretation of the grammar. We describe a tree as a set X of nodes and a set U of tuples $\langle m, d_1, ..., d_k \rangle$ with elements from X, satisfying conditions capturing tree geometry; m is interpreted as a mother node, and$d_1, ..., d_k$ as the ordered sequence of daughters of m.

Let $G = \langle N, T, W, \mathcal{L}, \mathcal{R}, s \rangle$ be a headed CFG, let $\langle X, U \rangle$ be a tree, and let ν be a function with domain X and range $\mathcal{N} \cup \mathcal{L}$. We write $\nu_1(x)$ for the first (category) component of $\nu(x)$, and $\nu_2(x)$ for the second (word) component; note that both terminal and non-terminal nodes have word and category components. Let r be a function defined on the non-terminal nodes of X, with range $ER(G)$; let l be a function defined on the non-terminal nodes of X, with range $EL(G)^*$. r labels non-terminals with rule events, and l labels non-terminals with sequences of lexical events interpretable as the lexical-choice events for the non-head daughters of the non-terminal. Licensing of a tree by a grammar is defined as follows.

Definition. The labeled tree $\langle X, U, \nu, r, l \rangle$ is licensed by G if and only if:

1. For each terminal node x in X the label $\nu(x)$ is an element of $\mathcal{L}$.
2. For each non-terminal node x in X the label $\nu(x)$ is an element of $\mathcal{N}$.
3. For any non-terminal node m, with daughters $d_1, ..., d_k$, $r(m)$ is a rule event $\langle \bar{w}, \bar{n}, \alpha, n, \beta \rangle$, where letting a be the length of α and b be the length of β:
   (a) $k = a + 1 + b$
   (b) $\nu(m) = \langle \bar{n}, \bar{w} \rangle$
   (c) $\alpha = c_1 ... c_{a-1}$, where this notation designates the empty sequence when $a = 0$
   (d) for each i, $1 \leq i \leq a$ $\nu_1(d_i) = c_1$.
   (e) $\nu(d_{a+1}) = \langle n, \bar{w} \rangle$
   (f) $\beta = c_{a+2} ... c_{a+b}$, where this notation designates the empty sequence when $b = 0$
   (g) for each i, $a + 1 \leq i \leq b$, $nu_1(d_i) = c_i$.

If in addition the category label of the root of the tree is the initial symbol, then we say that the tree is licensed as a sentence by G. Where $\tau = \langle X, U, \nu, r, l \rangle$ is a labeled tree licensed by G, we define $e(\tau)$ to be a function counting occurrences of events as labels in τ. The function has the following type:

$$e(\tau) : E(G) \to I\!N$$

Algebraically, we think of $e(\tau)$ as a monomial in the variables $E(G)$; the exponent of a given variable (or event) z is the number of occurrences of z in τ. We denote the evaluation of a polynomial or monomial ϕ in the variables $E(G)$ by subscripting: ϕ_p is the value of ϕ at the vector of reals p. For a monomial ϕ represented as a function from $E(G)$ to $I\!N$, evaluation is defined as follows.

$$\phi_p = \prod_{z \epsilon E(G)} p_z^{\phi(z)}$$

Relative to a parameter setting p, $[e(\tau)]_p$ is interpreted as the probabilistic weight of the labeled tree τ.[9]

We have so far omitted the details regarding lexicalization of the start symbol. For the lexicalized grammar we use a lexicalized start symbol, $\langle s, \text{START-WORD} \rangle$, where START-WORD is a dummy word not used elsewhere, and it behaves in a slightly non-obvious way. If τ is licensed by G and in addition the v maps the root of τ to $\langle s, w \rangle$, we say that τ is licensed as a sentence by G. N.B. w is not the dummy START-WORD. To account for the probability, we include another term in our monomial $e(\tau)$, namely the factor $p_{s,\text{START-WORD},s,w}$.

By the following construction, our lexicalized grammars can be mapped to ordinary PCFGs which license homomorphic trees, preserving the parameterization and the induced probability weighting. Let $\langle N, T, W, \mathcal{L}, \mathcal{R}, s \rangle$ be a a head-lexicalized CFG. For each rule $r \in \mathcal{R}$ we produce a set of PCFG rules, one for each possible lexicalization of r. The mother categories (left-hand sides) for the new rules are $\langle \bar{n}, \bar{w} \rangle$, where $\bar{n}$ is the mother of r and $\langle \bar{n}, \bar{w} \rangle \in \mathcal{N}$. On the right-hand side, we lexicalize the head differently depending on whether original head category was a terminal or non-terminal. Terminals are replaced by the word $\bar{w}$, whereas non-terminals are replaced with the tuple $\langle n, \bar{w} \rangle$. For each non-head category y in α or β in the original rule, we substitute $\langle y, \bar{w}, \bar{n} \rangle$. In other words, we encode the lexicalized head into each daughter category. For each new daughter category, we create additional CFG rules $\langle y, \bar{w}, \bar{n} \rangle \to \langle y, w \rangle$ for all $\langle y, w \rangle \in \mathcal{N}$. This procedure reconstructs our lexical choice events as rule events in the PCFG.

2.4 Parameter estimation

We work with a fixed grammar G; the inductive problem is posed as one of estimating a head-lexicalized PCFG with signature G. We use the standard method for estimating PCFGs based on the EM algorithm, with expectations computed by the inside-outside algorithm in a parse forest representation of possible analyses. Since the approach is familiar, we confine ourselves to reviewing the mathematical definition of the expectation step. The core notion is the event count monomial $c(\sigma, p)$ for a sentence or corpus σ; this is a function mapping events to non-negative real numbers:

$$c(\sigma, p) : E(G) \to I\!R^+$$

Before defining the count monomial for a sentence, we define event polynomials for sentences and corpora. The yield $y(\tau)$ of a labeled tree τ is defined recursively: the yield of a terminal node x is $\nu_2(x)$, or the word w. The yield of a non-terminal node is the concatenation of the

[9] As with ordinary PCFGs, caution is required regarding the sample space for the probability measure which is defined for finite trees by this construction, and the domain of the probability measure. Depending on p, the construction may or may not define a probability measure on the set of finite trees licensed by G. For the general case, infinite trees can be included in the sample space: infinite labeled trees are labeled trees with an infinite node set X. This requires an extension in the definition of the measure (not all subsets of the sample space are measurable) but does not affect the probabilities of finite trees. Booth and Thompson (1973) analyzes the conditions under which a probability measure over finite trees is defined.

yields of the ordered daughters of the node. A *sentence* is a finite sequence of words, i.e. an element of W^+. We have already defined the event monomial $e(\tau)$ of a tree licensed by a head-lexicalized PCFG p. The event polynomial for a sentence σ is the polynomial sum of the event monomials $e(\tau)$ of all trees τ licensed by G and having yield σ.[10] A *corpus* is a finite sequence of sentences, that is an element of $(W^+)^+$. The event polynomial $e(\mathcal{C})$ of a corpus $\mathcal{C}$ is the polynomial product of the event polynomials for its component sentences. Fixing a grammar G and a corpus $\mathcal{C}$, the optimization problem we seek to solve is to find a parameter vector p with signature G such that the probabilistic weight $[e(\mathcal{C})]_p$ of the corpus is as high as possible.

The algorithm of Baum and Sell (1968) for constrained maximization of polynomials involves computing an event count monomial determined by a given polynomial ϕ and parameter setting p; in our application, ϕ is the corpus event polynomial e. Our algorithm works sentence by sentence, and we will define $c(\sigma, p)$ for a sentence σ directly in terms of the event monomials for trees τ with yield σ. The sum in the first equation below is a pointwise sum of vectors indexed by $E(G)$; juxtaposition of the fraction with $e(\tau)$ represents a scalar product.

(13) $c(\sigma, p) = \sum_{\tau|y(\tau)=\sigma} \frac{[e(\tau)]_p}{[e(\sigma)]_p} e(\tau)$

(14) $[e(\sigma)]_p = \sum_{\tau|y(\tau)=\sigma} [e(\tau)]_p$

The second equation can be used (though in this form not efficiently) to calculate the probabilistic weight of a sentence σ; it is a consequence of the fact that evaluation of polynomials is a homomorphism from the ring of polynomials to the reals (e.g.Lang (1993)).

Where z is an event, $c(\sigma, p)(z)$ has the probabilistic interpretation of the expected number of occurrences of the the event z in a tree with yield σ. The sample space consists of labeled trees, with probability measure parameterized by p, as defined above for finite trees; the random variable for which a conditional expectation is computed maps a tree τ to $e(\tau)(z)$. The conditioning event for the conditional expectation is that $y(\tau) = \sigma$.

Algorithmically, we compute the event count for a sentence σ with the inside-outside algorithm in a parse forest representation of the set of tree analyses of σ. A parse forest is an and-or graph representation of a set of tree analyses (Lang 1989), in which event counts can be computed efficiently using the inside-outside algorithm (Baker 1979).

Given a parameter setting p, event counts are computed and summed over the sentences in the corpus. Using a suitable definition of the event-count monomial determined by a polynomial, this procedure can be shown to compute the event-count monomial for the corpus. In the algorithm of Baum and Sell (1968) new parameter values would be defined as relative frequencies of event counts, equivalent to maximum-likelihood estimation. We use instead a smoothing scheme in order to deal with the size of the parameter space and the resulting problems that (i) counts are zero for the majority of events, and (ii) the parameter space is too large to be represented directly in computer memory. The procedure can be conceptualized as a reparameterization of the model where additional smoothing parameters are defined in terms of event counts.

[10] Since we defined trees in terms of node carrier sets X, here "all trees" is to be understood as all trees, up to isomorphism.

Smoothing of rule distributions

Lexicalized rules are smoothed against a non-lexicalized grammar in a backing-off scheme (Katz 1980). Recall that a lexicalized rule event has the form $\langle \bar{w}, \bar{n}, \alpha, n, \beta \rangle$ where $\langle \bar{n}, \bar{w} \rangle$ is the lexicalized mother category. We define the event count for a lexicalized mother category by summation over right hand sides:

$$c(\langle \bar{n}, \bar{w} \rangle) = \sum_{\alpha,n,\beta} c(\langle \bar{w}, \bar{n}, \alpha, n, \beta \rangle)$$

The summation ranges over right hand sides α, n, β such that $\langle \bar{n}, \alpha, n, \beta \rangle$ is a rule. The unsmoothed probability for the rule event $\langle \langle \bar{n}, w \rangle, \alpha, n, \beta \rangle$ is defined as a relative frequency of counts:

$$p_{\bar{w},\bar{n},\alpha,n,\beta} = \frac{c(\langle \bar{w}, \bar{n}, \alpha, n, \beta \rangle)}{c(\langle \bar{n}, w \rangle)}$$

Unlexicalized rule counts $\bar{c}$ are defined by summing over the word dimension in the lexicalized count:

$$\bar{c}(\langle \bar{n}, \alpha, n, \beta \rangle) = \sum_{\bar{w}} c(\langle \bar{w}, \bar{n}, \alpha, n, \beta \rangle)$$

From this we define an unlexicalized rule distribution $p_{c,\alpha,n,\beta}$, as a relative frequency of counts:

$$p_{\bar{n},\alpha,n,\beta} = \frac{\bar{c}(\langle \bar{n}, \alpha, n, \beta \rangle)}{\bar{c}(\bar{n})}$$

$\bar{c}(\bar{n})$ is defined in a way analogous to $\bar{c}(\langle \bar{n}, w \rangle)$. To smooth the lexicalized and unlexicalized distributions together, we use a weighting parameter dependent on the corpus frequency $f(w)$ of a word w.

$$\begin{aligned} \tilde{p}_{\bar{w},\bar{n},\alpha,n,\beta} = \;& \lambda(c(\bar{w})) p_{\bar{w},\bar{n},\alpha,n,\beta} \\ & +(1 - \lambda(c(\bar{w}))) p_{\bar{n},\alpha,n,\beta} \end{aligned}$$

where $0 \leq \lambda(f(w)) \leq 1$. The dependency of the smoothing parameter on word frequency allows the lexicalized distribution p' to be assigned greater weight for frequent words, and less weight for infrequent words. The actual values are chosen by treating the two distributions as components of a small set of trivial hidden Markov Models (HMMs), indexed by word frequency. By training the HMMs with the forward-backward algorithm, we tune the λs to optimal values in terms of cross entropy. The actual λs and frequency ranges are given below.

freq cut-off	0.1	4.0	8.0	16.0	∞
$\lambda(f)$	0.0	0.05	0.55	0.85	0.95

Smoothing of word distributions

For the lexical choice distributions, we use an absolute discounting scheme from Ney et al. (1994). Recall that lexical choice events are of the form $\langle \bar{w}, \bar{n}, n, w \rangle$, where $\bar{n}$ is interpreted as the mother category, $\bar{w}$ is interpreted as the mother word, n is interpreted as the (non-head) daughter category, and w is interpreted as the chosen daughter word. The choice distributions $p_{w',\bar{n},n,w}$ realize a set of conditional probability distributions, more conventionally written $p(w|\bar{w}, \bar{n}, n)$. For each conditioning triple $\langle \bar{w}, \bar{n}, n \rangle$, we compute a discount $D_{\bar{w},\bar{n},n}$ which is used to decrement the counts for events of the form $\langle \bar{w}, \bar{n}, n, w \rangle$.

The discounted count is defined by:

$$\hat{c}(\langle \bar{w}, \bar{n}, n, w \rangle) = max(0, c(\langle \bar{w}, \bar{n}, n, w \rangle) - D_{\bar{w},\bar{n},n})$$

where the maximization sets counts which are below the discount factor to 0.

The smoothed distribution is then defined by:

$$\tilde{p}(w|\bar{w}, \bar{n}, n) = \frac{\hat{c}_{\langle \bar{w},\bar{n},n \rangle}(w)}{\sum_w c_{\langle \bar{w},\bar{n},n \rangle}(w)} + k_{\langle \bar{w},\bar{n},n \rangle} \tilde{p}(w|\bar{n}, n)$$

where $k_{\langle \bar{w},\bar{n},n \rangle}$ is the normalization constant

$$k_{\langle \bar{w},\bar{n},n \rangle} = \frac{\sum_w \min(c_{\langle \bar{w},\bar{n},n \rangle}(w), D_{\langle \bar{w},\bar{n},n \rangle})}{\sum_w c_{\langle \bar{w},\bar{n},n \rangle}(w)}$$

and $\tilde{p}(w|\bar{n}, n)$ is a smoother distribution over w, about which we will have more to say later.

The discount factor is chosen according to the leave-one-out principle.[11] The idea is to simulate test circumstances by withholding part of the training data and scoring the model estimated from this incomplete training data on the held-out data. Let O be a set of observations, s a smoothing parameter, $p_{O,s}$ the probability distribution estimated from O and s. The scoring function for a smoothed distribution is the log-likelihood it assigns to the held-out data. To make efficient use of the training data, we restrict the held-out data to a single observation, which is allowed to vary across the data set. Thus, the desired smoothing parameter value is:

$$\max_s \sum_{o \in O} \log p_{(O-o,s)}(o)$$

There is, in general, no closed form for the optimal s. However, Ney gives an approximation and a procedure for iteratively refining the approximation under conditions met by our lexical choice distributions.

The standard formula for D is in terms of $N(i)$, defined as

$$N(i) = \textstyle\sum_{w|c(w)=i} 1 \quad i > 0$$

which is the number of event types which occur exactly i times. This assumes events are completely observable, so that the event counts are restricted to the natural numbers. Our events are only partially observable, and so may take on any non-negative real value. We define $R(i)$ to be the natural extension of $N(i)$ to real-valued observations:

$$R(i) = \sum_{w|i-0.5<=c(w)<i+0.5} 1$$

The approximate discount factor $\widehat{D}_{\langle \bar{w},\bar{n},n \rangle}$ is then defined as:

$$\widehat{D}_{\langle \bar{w},\bar{n},n \rangle} = \begin{cases} \frac{R(1)}{R(1)+2R(2)} & R(1), R(2) > 0, \\ & \sum_i iR(i) > 10 \\ D'(\approx 0.8) & \text{otherwise} \end{cases}$$

We use a default discount value, D', to handle the cases where there is not enough data to estimate a smoothing parameter. This represents a compromise between estimating a discount by pooling all counts, which leads to cruder smoothing, and estimating a separate smoothing parameter for each distribution, which degenerates when there is not enough data in to estimate the smoothing parameter. $\acute{D}$ is estimated by pooling all tuples not directly estimated above, and applying the adapted discount formula to the pooled data. Let F be the set of all events failing all the 'if' condition above, i.e., either $R(1) = 0$ or $R(2) = 0$ or $\sum_i iR(i) > 10$. Let $R_F(i)$ be the number of event types in F which occur between $i - 0.5$ and $i + 0.5$ times. Then D' is:

$$D' = \frac{R_F(1)}{R_F(1) + 2R_F(2)}$$

As Ney's results suggest that the approximation is very good in terms of its effects on overall entropy, we do not iteratively refine the discount factors.

[11] Leave-one-out is related to the *held out* method (Duda and Hart 1973) and n-way cross-validation (Wahba and Wold 1975).

We now return to the back-off distribution $\tilde{p}(w|\bar{n}, n)$. As appropriate for a back-off distribution, $\tilde{p}(w|\bar{n}, n)$ has fewer parameters, allowing for more accurate parameter estimation and fewer sparse data problems. However, our model faces the problem of rare and even novel words, to which it must still assign non-zero probabilities. We therefore take the standard step of cascading the back-off distribution(s):

$$\tilde{p}(w|\bar{n}, n) = \frac{\hat{c}_{\langle \bar{n},n\rangle}(w)}{\sum_w c_{\langle \bar{n},n\rangle}(w)} + k_{\langle \bar{n},n\rangle}\tilde{p}(w|n)$$

$$\tilde{p}(w|n) = \frac{\hat{c}_{\langle d\rangle}(w)}{\sum_w c_{\langle d\rangle}(w)} + k_{\langle d\rangle}\tilde{p}(w)$$

$$\tilde{p}(w) = \frac{\hat{c}(w)}{\sum_w c(w)} + k\rho(c \in w)$$

where $\rho(c \in w)$ is a poisson distribution over character strings. $\hat{c}_{\langle \bar{n},n\rangle}(w) = \sum_{\bar{w}} c_{\langle \bar{w},\bar{n},n\rangle}(w) - D_{\langle \bar{n},n\rangle}$. That is, we compute the discount from the pooled counts. The discount factors and normalization constants are computed, mutatis mutandis, as before.

Smoothing across grammatical categories

The trigram state model introduces a large number of categories, with the unfortunate effect that at this level the lexical dependencies are overparameterized. Consider the hypothetical observation

$$\langle \text{wrote}, \text{NP:VFP}, \text{VFC1:NP}, \text{book}\rangle$$

indicating that *book* was seen as a (direct) object of *wrote*, and *wrote* was preceded by a noun chunk. If some other category occurs before the verb, say THAT, this would be counted as a distinct observation, and the two would fall into separate distributions. To add further injury, the compound categories reduce the effectiveness of smoothing and divorce the collocates identified with the trigram model from those found with conventional complementation rules. The penultimate smoothed distribution for the observation above is $\tilde{p}(\text{book}|\text{VFP1:NP})$, rather than the much more intuitive $\tilde{p}(\text{book}|\text{NP})$. The implication is that, until we reach the unconditioned string model, there is no sharing of data between the trigam and conventional rules.

To alleviate this problem, we map the trigram states onto their most recent (right-hand) categories. This transforms our hypothetical example above into

$$\langle \text{wrote}, \text{VFP1}, \text{NP}, \text{book}\rangle$$

This is identical to the observation that would be recorded with the rule VFP → VFC′ NP; the data can be pooled at every granularity.

prob	freq	word	prob	freq	word
0.043	352.9	need	0.016	129.6	chance
0.039	318.8	attempt	0.013	102.6	tendency
0.038	314.6	ability	0.011	93.6	decision
0.033	266.7	time *	0.010	83.3	something *
0.027	222.7	way *	0.0098	80.6	capacity
0.024	195.7	opportunity	0.0097	79.4	desire
0.023	187.8	right	0.0088	72.5	effort
0.020	167.0	nothing *	0.0087	71.5	thing *
0.020	163.3	power	0.0086	70.2	people
0.018	143.9	failure	0.0083	68.1	evidence

Table 2.1: Nouns selecting an infinitival complement/adjunct

2.5 An experiment

We estimated a head-lexicalized PCFG from parts of the British National Corpus, using the grammar described in section 1 and the estimation method of the previous section. A bootstrapping method was used, in which first a non-lexicalized probabilistic model was used to collect lexicalized event counts. On the next iteration, counts were summed based on a lexicalized weighting of parses, as described in the previous section.

We restricted the analyses considered to those consistent with the part of speech tags specified in the BNC, which are output of the CLAWS tagger. In some cases the BNC specifies a disjunction of tags, and the mapping from BNC tags to our own terminal categories introduced further ambiguity. For instance, unlike the BNC, we use distinct parts of speech for passive participles and perfect participles. We also collapse some distinctions made in the BNC; for instance, we use a single part of speech tag for finite verb forms, collapsing present and past tenses.

In each lexicalized iteration, event counts were collected over a contiguous five million word segment of the corpus. Parameters were re-computed according to the formulas described above, and the procedure was iterated on the next contiguous five-million word segment.

We report estimated frequencies obtained after eight lexicalized iterations. A useful informal way of examining the induced valence lexicon is to rank heads which select a given frame by frequency of their using the frame. In table 2.5, we look at nouns selecting a to-infinitive.

The frequency column lists the expected frequency in a fixed 5-million word corpus segment of the given nominal head (e.g. *ability* in the context of a VTOP complement. The expected frequency is computed using the method described above. The first column is derived from the second; it gives unsmoothed probabilities for the head nominal word given the local phrase structure configuration.

There is a proviso about the lexical relations being identified. While the unstarred nouns select what under most analyses would be considered a complement, the selected element in the case of *time, way, nothing, something, place* and *thing* is an infinitival relative clause. This is reflected in paraphrasability with a modalized finite relative clause, or an infinitival relative clause with an overt wh phrase:[12]

(15) a. The main thing to realise with trailer driving is that it only takes one mistake to wreck the trailer.
b. The main thing which one should realize with trailer driving is ...
c. the best way to find out what is available
d. the best way in which to find out what is available

Our syntactic categories do not distinguish infinitival phrases (category VTOP) containing free traces from ones not containing free traces. In this sense, the syntactic model is not capable of distinguishing infinitival relative clauses from other infinitival phrases. Independently, our model does not have the means of distinguishing complements from lexically conditioned or otherwise frequent adjuncts.[13] If a certain adjunct category occurs frequently with a given head, an association will be learned just as with a complement. Thus—assuming that of the complement/adjunct distinction is valid, something which we do not take for granted—the model described here learns lexically conditioned complements and adjuncts, rather than complements alone. We do not regard this as a defect, for two reasons. First, for some purposes, the distinction between complements and adjuncts is of no significance. For other purposes (for instance in the theory of lexical semantic structure, and in certain areas of syntax) the distinction may be significant, but having a good account of conditioned adjuncts is just as important as having a good account of complements. In this sense, our procedure is learning only part of a lexical representation; it does not learn whatever features characterize the complement/adjunct distinction. Many other features of lexical representations have the same status; for instance the procedure in present form does not learn control features.

Table 2.2 lists verbs occurring with high estimated frequency with an NP plus to-infinitive

[12] (15a) and (15c) are quoted from the BNC.

[13] By lexically conditioned, we mean that the adjunct can combine with a fairly confined class of modified phrases, and that the integration of the semantics of the adjunct is closely tied to lexically-conditioned factors in the semantics of the modified phrase. This is true for instance of path prepositional phrases in theories where such phrases are considered adjuncts, e.g. with motion or perception verbs (*John walked across the field, John looked across the field*). This brings up the point that theoretical work draws the line between arguments and adjuncts in a position different from what is dictated by naive considerations. For instance, in the theory of Grimshaw (1990), only a class of nouns having lexical representations involving complex event structure take arguments; an example is *expression* in the sentence *the frequent expression of one's feelings is desirable.* Other nouns, including for instance the relational noun *member* in *a member of the commmittee*, take adjuncts rather than arguments.

prob	freq	word	
0.06456	390.6	is	is the first to acknowledge
0.03602	217.9	have	have a great deal to offer
0.03030	183.3	allow	
0.02584	156.3	want	
0.02034	123.0	use	use a torch to see
0.01603	96.98	enable	
0.01574	95.23	asked	
0.01562	94.52	wanted	
0.01419	85.86	take	take you to meet Brian
0.01332	80.57	ask	
0.01211	73.26	led	
0.01157	70.03	expect	

Table 2.2: Verbs combining with an NP and an infinitive

complement frame. In this table, to avoid redundancy, we include only the first occurrence of an inflectional form of a given lemma. For cases where we feel the NP-VTOP frame is not intuitively obvious, we include an example.

Finally, table 2.3 lists verbs taking the frame NP PARTP, a noun phrase followed by a particle.

These data are positive, in the sense that the learned frames are correct according to our criteria. However, this way of looking at results does not bear very directly on the specific goal of the learning experiment, which has to do with the distribution over subcategorization frames conditioned by a given head. In section 2.6, we will measure the extent to which such distributions have been learned.

Word-word relations

Lexical events $\langle \bar{w}, \bar{n}, n, w \rangle$ are used to choose the heads of non-head daughters. The conditioning factors are the mother category $\bar{n}$, the mother head $\bar{n}$, and the daughter category n. These events model frequent head-head pairs occurring a modification of complementation relation. In table 2.4 we list the eight most frequent adjective modifiers of *road*, the eight most frequent adverb modifiers of *satisfactory*, and the eight most common object heads for the verb *address*.

As described above, such parameters are recorded only for words which have been observed with an estimated frequency above a cutoff. Following each head above, we indicate in parentheses how many parameters are recorded for a 5 million and a 30 million word model.

prob	freq	word
0.0551072	96.9494	put
0.0326008	57.3542	take
0.0245983	43.2756	get
0.0211335	37.1799	let
0.0208253	36.6378	is
0.0201579	35.4635	took
0.017727	31.1868	brought
0.017212	30.2808	turned
0.0166642	29.3172	have
0.0142207	25.0183	picked
0.011036	19.4155	pulled
0.00963676	16.9538	set

Table 2.3: Verbs combining with a noun phrase followed by a particle

The lexical selection model in effect threads a word bigram model along the structure of heads marked in the syntax. Each lexical head (other than the topmost one) is conditioned on some other lexical head. This allows the model to capture a wide range of collocational dependencies. Consider the italicized range of words in the following sentence drawn from the BNC.

(16) PASS & CO. *have over thirty years experience* in prevention and preservation work in houses

The following collocates are all relevant to evaluating the correct parse: *have experience* (verb-object head); *over thirty* (adverb-cardinal); *thirty years* (cardinal-noun); *years experience* (noun noun). All of these are frequent enough to be captured in the model we have estimated from 30 million words of the BNC.

2.6 Frame evaluation

In this section, we suggest an evaluation a probabilistic complementation frame, and apply it to the distributions for three specific verbs.

We obtained frame frequency counts for the verbs *allows*, *reached*, and *prove* by hand-classifying one to two hundred occurrences of each. Some observed frames were not present in the grammar, for predictable reasons. Frames such as SBAR require high-level constructs not present in the current grammar. Unusual and unorthodox frames turned up, e.g. PART PP PP. Third, some ordinary frames were simply overlooked. In the tables below we indicate absent frames by enclosing the "true", hand-classified figure in parends, and putting a dash in the column for the "observed", model-classified figure. It is a straightforward matter to deduce how the model should classify frame occurrences for which the correct rule is absent. For this reason, we report results both for the *complete* set of frames, the direct results of hand-classification, and the *feasible* set of frames, the reduced set of frames which the model has to choose from. The column headed "true" indicates counts that differ between the two frame sets by putting the *complete* counts inside parends, and the *feasible* counts outside. The tables also indicate the intrinsic frame entropy for each word, given both the complete and feasible frame sets, again putting the *complete* figureres in parends.

road (63/244)		satisfactory (14/37)		address (24/86)	
ADJ	prob	ADV	prob	NN	prob
main	0.29	entirely	0.17	question	0.086
roman	0.062	highly	0.11	issue	0.086
new	0.055	most	0.09	themselves	0.059
high	0.051	very	0.075	issues	0.031
old	0.030	quite	0.055	structure	0.031
narrow	0.017	wholly	0.032	argument	0.014
important	0.011	uncommonly	0.0037	questions	0.0043
modern	0.0091	especially	0.0037	electorate	0.0043

Table 2.4: Adjectives selected by *road*, adverbs selected by *satisfactory*, and object nouns selected by the verb *address*. In parends we give the number of entries deemed worthwhile in 5M and 30M word corpus segments.

(17)

allows entropy: 1.459 (1.929)

freq		
true	obs	frame
135	116.5	NP VTOP
40	35.4	NP
11	9.2	PP
7(5)	13.5	<NIL>
4	10.0	NP NP
3	9.4	NP PP
(2)	–	SBAR

(18)

reached entropy: 2.140 (2.428)

freq		
true	obs	frame
117	97.3	NP
26(21)	14.7	NP PP
11	14.1	PP
9(8)	6.3	PART PP
(5)	–	NP PP PP
4	1.9	NP NP
3(2)	3.6	PART
2	21.8	<NIL >
2	1.6	PP PP
(1)	–	PART PP PP
(1)	–	PART VTOP

(19)

word: *proved* entropy: 1.616 (1.748)

freq		
true	obs	frame
72	60.6	AP
37	34.9	VTOP
(15)	–	SBAR
6	6.9	NP PP
5	2.7	NP AP
4	1.6	NP NP
4	2.3	NP VTOP
(3)	–	S
(2)	–	PP SBAR
2	4.1	PP
1	35.0	<NIL >
1	0	AP PP

For a fixed verb, we represent such a frequency count by $f(\alpha)$, where α is the frame; we represent the frame probability distribution estimated in our experiment by p_α. The idea is to measure the agreement between the model's distributions and the observed frequency counts. Our measure is based on the information-theoretic notion of cross-entropy. Letting $N = \sum_\alpha f(\alpha)$, the total number of frame occurrences counted, we define the per-occurrence cross-entropy as follows:

$$\sum_\alpha - \log p_\alpha \times \frac{f(\alpha)}{N}$$

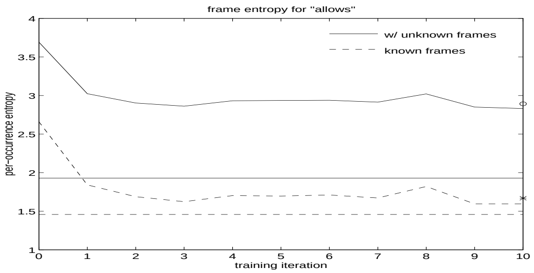


Figure 2.8: Per-occurrence frame cross-entropy.

The term $-\log p_\alpha$ can be conceptualized as the number of bits required to designate frame α in an ideal code based on the probability distribution p, while $\frac{f(\alpha)}{N}$ scales this by the observed relative frequency of α. Disagreement between p and the real distribution of frames has the effect of raising the cross entropy.

For the cross-entropy formula to be informative, p_α must be non-zero whenever $f(\alpha)$ is. Because some of the actual frames were not included in the present generation of our grammar this is not the case for our rule distributions, even smoothed against the unlexicalized rules. We dealt with this instead by smoothing against a a spelling model, a poisson distribution over categories which assigns non-zero probability to all frames, observed or not. In the case of *proved*, unknown frames involving SBAR are fairly frequent, and the spelling model produces quite a high penalty.

The cross-entropy measure has three features to recommend it. First, it provides a theoretically motivated price measure for various bits of information. The price for missing an infrequent event is low; more common events cost more, in terms of their entropy penalty. Second, by measuring the entropy of the distribution, one can separate the difficulty of the distribution to be learned from what is actually learned. Simply identifying a verb's frames precludes measuring whether the task is easy or difficult. Third, it avoids relying on other sources of information, such as dictionaries, which may be unreliable or incompatible. In particular, one must either reach consensus with the outside source on the argument/adjunct distinction, or somehow map the learned frames to the outside frame scheme.

Figures 2.8 through 2.10 show the sample cross entropy. The horizontal axis indicates training iteration for the probability models, with 0 being the bootstrap model estimated from an unlexicalized grammar. In addition, we indicate the cross entropy for the combined model on the right-hand edge of each graph, using an "o" and a "*" for the complete and feasible frame sets, respectively. The horizontal lines indicate the entropy of the observed frame distribution; optimal model behavior should approach this line. The difference between

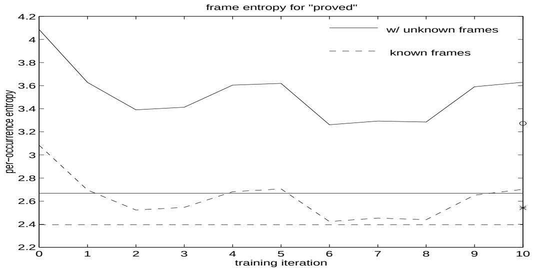


Figure 2.9: Per-occurrence frame cross-entropy.

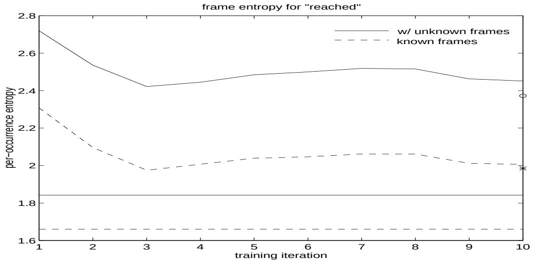


Figure 2.10: Per-occurrence frame cross-entropy.

	allows		proved		reached	
	feasible	complete	feasible	complete	feasible	complete
entropy	1.459	1.929	2.397	2.668	1.660	1.842
revised model c.e.	1.628	–	2.438	–	1.787	–
combined model c.e.	1.664	2.893	2.542	3.273	2.019	2.467

Figure 2.11: Comparison of optimal performance (entropy), combined model performance, and performance of the revised model trained on the test sample.

the two reflects the ***relative entropy***, or the number of additional bits needed to guess the correct frame, when using the model's distribution.

A visual inspection of the graphs shows that the cross entropy for the complete and feasible frame sets closely track each other. The complete set exacts an entropy penalty of various size, depending on the frequency of the unknown frames, but there is no tendency to converge on the feasible frames at the expense of the correct ones.

The major trend for all words is that cross entropy drops in the initial iterations of lexicalized parsing, up to around the third training iteration. Thereafter the movement is erratic. Results for the combined model are mediocre, better than the worst performance but worse than the best performance seen for individual iterations.

The erratic performance of the individual training models and the lacklustre performance of the combined model suggest that either the model is failing to learn or the 5 million word segment size is too small for the target distributions, and sample variance is causing fluctuation in the model distributions. We will argue that neither of these is the case, but rather are evidence that the heterogeneous content of the BNC produces a non-stationary distribution.

To confirm that the model learns correctly, we carried out the following mini-experiment. Using the combined model, we estimated the frame observations over the hand-evaluated test sample. Using only those observations, we re-estimated the model distributions for the target words, and re-evaluated the cross entropy on the test sample. In all cases, the re-estimated model showed a large jump in entropy, measured on the set of feasible frames[14] We tabulate these results in table 2.6, where the re-estimated model is designated as "revised".

The BNC is a "balanced" corpus, meaning it contains excerpts drawn from a wide variety of genres, including spoken dialogues, pulp fiction, political pamphlets, and more. The average sample length is about 25,000 words. A 5 million word sample therefore contains about 200 different sample texts, which seemed enough to overcome the presumed discrepancies between

[14] Since the complete frames cannot be completely converged on, measuring the relative entropy of the complete frames and the model is less informative.

the distributions in the different texts. Instead it appears that the distributions are sharper than we anticipated, and hence 200 samples is not enough to overcome the variance, or the organization of the BNC, with related materials grouped contiguously, has defeated our attempt to "homogenize" the distribution. The tantalizing suggestion of the data is that there is a very good reason why other researchers, also working on lexicon extraction from text, have reported both finding entries missing from lexicons and lexicon entries that did not appear, even in large text samples: the reason is simply that lexicon builders have overestimated the stability and uniformity of the distributions. The second suggestion is that static lexica are likely to be of only limited utility. Either one can make very crude lexica, with a sharply reduced set of frames, likely to appear across a wide variety of texts, or one can include a great many frames, most of which will not appear in any particular sample text. Automatic methods of acquiring, tuning, or somehow conditioning the lexical entries seem the order of the day.

2.7 Optimal parses

Although identifying a unique parse does not play a role in our experiment, it is potentially useful in a broad variety of applications, including machine translation, bilingual text alignment, and speech generation. A simple criterion is to pick a parse with maximal probability; this is identified in a parse forest by iterating from terminal nodes, multiplying daughter probabilities and the local node weight at *and*-nodes, and choosing a daughter with maximal probability at *or*-nodes. In table 2.5 we give several examples of maximal probability parses. Entries in the left column are maximal phrasal categories. Successive phrases at this level are handled by the state grammar. Below this level, we indicate structure with indentation.[15] Note that two PPs in the first example are handled by the state model, rather than being attached to the appropriate noun phrases. In the second example, the tensed passive verb phrase is chopped up by an interjection.

Other optimality criteria can be defined. The structure on noun chunks is often highly ambiguous, because of bracketing and part of speech ambiguities among modifiers. For many purposes, the internal structure of an noun chunk is irrelevant; one just wants to identify the chunk. In what we call a pseudo-Viterbi parse, probabilities are summed within chunks by the inside algorithm. Above the chunk level, a highest-probability tree is computed, as described above.

[15] PPs have been adjusted to hang off NP rather than N1, in conformity with the grammar described above, although these parses were obtained with a grammar which attaches PPs to N1.

```
PP  at PREP
    NC its PRO\$ next ADJ meeting NSG
NP the DETSG group NSG
VFP VFC received VF
    NC a DETSG paper NSG
    PP PC from PREP
       NP joyce PN skinner PN
PP  on PREP
    NP a DETSG possible ADJ college NSG
      PP PC of
         NC PREP education NSG model NSG
COMP  -- COM
NP   one CD
PP   of PREP
        NP several DETPL models NPL
COMP    , COM
PP   including PREP
     NP a DETSG university NSG model NSG
PP   by PREP
     NP professor PN john PN ziman PN
PERP . PER

NP    stewart PN   mason PN
COMP  , COM
NP    the DETSG new ADJ chairman NSG
      PP of PREP
         NP the DETSG ncdad NSG
COMP   , COM
VFP      was VBF
COMP     -- COM
SUBP     as SUB
NP       we PROSG
VFP      have VHF seen VN
COMP     -- COM
VPASSP   convinced VPASS
         PP by PREP
            NC the DETSG early ADV 1970s CD
THATP    that THAT
NP       a DETSG merger NSG
VFP      was VBF both ADV inevitable ADJ
                  and CONJ desirable ADJ
PERP     . PER
```

Table 2.5: Maximal-probability parses

2.7.1 Notes on the implementation, parsing times, etc

The parser used for the bootstrap phase is a vanilla CFG chart parser, operating bottom-up with top-down predictive filtering. Chart entries are assigned probabilities using the unlexicalized PCFG and the lexicalized frequencies are found by carrying out a modified Inside-Outside algorithm which simulates lexicalization of the chart.

In the iterative training phase, an unlexicalized context-free skeleton is found with the same parser. We transform this into its lexicalized form–categories become $\langle w, n \rangle$ pairs and rules acquire lexical heads–and carry out the standard Inside-Outside using the more elaborate head-lexicalized PCFG model. Average speed of the parser during iterative training, including parsing, probability calculation, and recording observations, is 10.4 words per second on a Sun Sparc-20. The memory requirements for a model generated from a 5M word segment are about 90Mbyte. The upshot of all this is that we can train about 1M words per day on one machine, and a single 5M word iteration requires one machine work week.

Note that the trigram state model, while linguistically unmotivated, will nonetheless yield many appropriate collocates in the word-word frequency counts. This has been the none-too-secret ingredient in the success of n-gram models in speech recognition. Our model is a variation on these, in that it includes grammatical categories in the conditioning parameters. This increases the size of our bigram model to approximately the same size, in terms of parameters, as a more ordinary trigram model. In terms of model entropy, category information is presumably not as effective as an additional word for conditioning. However, we enjoy an advantage over traditional n-gram models in that we structure the lexical dependencies with our syntactic rules. Hence, at the state machine level, dependencies will generally be between heads, even if they lie more than two words apart. Within phrases, we have even better information, on average, and we restrict our attention to the probabilistic relation between heads and their dependents.

2.8 Conclusion

Lexicalization both necessary and sufficient for frame extraction. The lexicalization may, in fact, be too sensitive to the data, as suggested by the noisy behavior of the per-frame-occurrence cross-entropy. If so, this suggests an interesting limit to what can be stated probabilistically.

Automatic Semantic Classification of Verbs According to their Alternation Behaviour[*]

Sabine Schulte im Walde

Abstract

An automatic semantic classification of verbs was performed by first determining the verbs' alternation behaviour and then clustering the verbs on that basis. The alternation behaviour of the verbs was outlined by inducing syntactic subcategorisation frames from maximum probability (Viterbi) parses of a robust statistical parser, completed by assigning WordNet classes to the frames' arguments. The clustering was achieved (a) iteratively by measuring the relative entropy between the verbs' probability distributions over the different types of frames, and (b) by utilising a latent class analysis based on the joint frequencies of verbs and frame types. Using Levin's verb classification (Levin 1993) as evaluation basis, (a) 61% and (b) 54% of the verbs were classified correctly into semantic classes.

[*] The work was performed within the scope of my diploma thesis (Schulte im Walde 1998).

3.1 Motivation

For this work, I assumed that the diathesis alternation of verbs, i.e. the alternation in the expression of the verbs' arguments, is a basis for the comparison of the verbs' meanings. More specifically, I empirically investigated the proposition that verbs can be semantically classified according to their syntactic alternation behaviour concerning subcategorisation frames and their selectional preferences for the arguments within the frames.

The idea of a semantic classification according to alternation behaviour is related to Levin (Levin 1993) who defined verb classes on the basis of the verbs' alternation behaviour. Consider, for example, the semantic class of *Vehicle Names* containing verbs like *balloon, bicycle, canoe, skate, ski* because they agree in the following properties:

- Intransitive Use, possibly followed by a path:

  (20) a. They skated.
  b. They skated along the canal/across the lake.

- Induced Action Alternation (some verbs):

  a sub-type of Transitive Alternation, where the transitive use of the verb can be paraphrased as causing the action named by the verb; the causee is typically an animate volitional entity induced to act by the causer; the verb must be accompanied by a directional phrase

  (21) a. He skated Penny around the rink.
  b. Penny skated around the rink.

- Locative Preposition Drop Alternation (some verbs):

  (22) a. They skated along the canals.
  b. They skated the canals.

- Resultative Phrase:

  an XP which describes the state achieved by the referent of the noun phrase it is predicated of as a result of the action named by the verb

  Penny skated her skate blades blunt.

As Levin did, I attempted to derive verb classes from the verbs' behaviour. The information I fed into an automatic deduction process for semantic classes was thereby referring back to Chomsky's (Chomsky 1965) demands for the utterance of verbs: the verbs' behaviour was defined by their subcategorisation rules and their selectional rules.

Such a definition of the verb's semantic class can be considered as part of its lexical entry, next to idiosyncratic information: the semantic class generalises as a type definition over a range of syntactic and semantic properties, to support Natural Language Processing in various areas like lexicography (by the enrichment of lexical knowledge), word sense disambiguation (by the provision of context information provided by the semantic verb type), or parsing (by the generalisation from verb tokens to verb types and the resulting restriction of syntactic structures). Klavans and Kan (Klavans and Kan 1998), for example, discriminate documents by type and semantic properties of the verbs within the documents.

3.2 Automatic Acquisition of Semantic Verb Classes

I empirically investigated the verbs' behaviour and their meanings by automatically inferring semantic verb classes with the help of data-intensive methods working on data from a large corpus, and by applying statistical methods proved useful for NLP-tasks. The inference process contained three main steps:

1. The induction of subcategorisation frames for verbs from a large corpus
2. The definition of selectional preferences for the subcategorisation frames
3. The clustering of the verbs into semantic verb classes, on account of the verbs' behaviour as defined in steps 1 and 2

Following sections 3.2.1 to 3.2.3 present the methods used for the three steps and their realisation.

3.2.1 Induction of Subcategorisation Frames

Within the first step of inducing purely syntactic subcategorisation frames for verbs I used the robust statistical head-entity parser as described in Carroll and Rooth (Carroll and Rooth 1998) which utilises an English context-free grammar and a lexicalised probability model to produce parse forests for sentences, where each sub-tree is annotated with information about the lexical head and the probability. I parsed the heterogeneous *British National Corpus (BNC)* and extracted the maximum probability (Viterbi) parses from the parse forests, for a total of 5.5 million sentences.

Based on the maximum probability parses I determined the main verb and all its arguments as the sentences' subcategorisation frame tokens. For example, the frame token of the sentence *Nobody excelled him in that judgement* would be defined by

`act*excelled subj*nobody obj*him pp*in*judgement`,

describing the full (active) verb form and the subject, object and prepositional phrase arguments as determined by the English grammar, each accompanied by its lexical head, the prepositional phrase accompanied by its lexical head and the head noun of the sub-ordinated noun phrase. I finished the frame description by lemmatising the head information in the subcategorisation frames.

To generalise over the verbs' usage of subcategorisation frames, I defined as 88 frame types those frames which appeared at least 2,000 times in total in the BNC sentence parses, disregarding the lexical head information. For example, the most frequent frame type was the transitive frame `subj:obj`. On the basis of the frame types I collected information about the joint frequencies of the verbs in the BNC and the subcategorisation frames they appeared with. Appendix 1 gives a full list of the 88 subcategorisation frame types and an example for the joint frequencies.

3.2.2 Selectional Preferences for Subcategorisation Frames

The next step after inducing the subcategorisation frame types was to refine the information by identifying a preferential ordering on conceptual classes for the argument slots in the frames. The basis I could use for the selectional preferences was provided by the lexical heads in the frame tokens as determined in section 3.2.1, for example the nouns appearing in the object slot of the transitive frame for the verb *drink* included *coffee, milk, beer*, demanding a conceptual class like *beverage* for this argument slot.

I followed Resnik (Resnik 1993)/(Resnik 1997) who defined *selectional preference* as the amount of information a verb provides about its semantic argument classes. He utilised the WordNet taxonomy (Beckwith et al. 1991) for a probabilistic model capturing the co-occurrence behaviour of verbs and conceptual classes, where the conceptual classes were identified by WordNet synsets, sets of synonymous nouns within a semantic hierarchy. Referring to the above example, the three nouns *coffee, milk, beer* are in three different synsets – since they are no synonyms –, but are all sub-ordinated to the synset defined by *beverage, drink, potable*. The goal in this example would therefore be to determine the relevant synset as the most selectionally preferred synset for the object slot of the verb *drink*.

Redefined for my usage, the selectional preference of a verb v concerning a certain semantic class c within a subcategorisation frame slot s was determined by the association ass between verb and semantic class:

$$ass(v_s, c_s) =_{def} p(c_s|v_s) log \frac{p(c_s|v_s)}{p(c_s)} \tag{3.2}$$

with the probabilities estimated by maximum likelihood:

$$p(c_s|v_s) = \frac{f(v_s, c_s)}{f(v_s)} \tag{3.3}$$

$$p(c_s) = \frac{f(c_s)}{\sum_{c' \in class} f(c'_s)} = \frac{f(c_s)}{f(s)} \tag{3.4}$$

To facilitate the understanding of the equations I briefly interpret the relevant parts:

1. $f(v_s, c_s)$ was defined by how often a certain semantic class appeared in a certain frame slot of a verb's frame type.
2. $f(v_s)$ was defined by the frequency of a certain verb regarding a specific frame type, i.e. the joint frequency of verb and frame type as determined in section 3.2.1.
3. $f(c_s)$ was defined by how often a certain semantic class appeared in a certain frame slot of a frame type disregarding the verb.
4. $\sum_{c' \in class} f(c'_s)$ equals $f(s)$, the frequency of the argument slot within a certain frame type, since summing over all possible classes within a subcategorisation frame slot was equal to the number of times the slot appeared.
5. $f(s)$ was defined by the number of times the frame type appeared (as determined in section 3.2.1), since the frequency of a frame type equals the frequency of that frame with a certain slot marked.

The frequencies of a semantic class concerning an argument slot of a frame type (dependent or independent of a verb) were calculated by an approach slightly different to Resnik's, originally proposed by Ribas (Ribas 1994)/(Ribas 1995): for each noun appearing in a certain argument position its frequency was divided by the number of senses the noun was assigned by the WordNet hierarchy,[16] to display the uncertainty about the sense of the noun.[17] The fraction was given each conceptual class in the hierarchy to which the noun belonged and projected upwards until a top node was reached. The result was a numerical distribution over the WordNet classes:

$$f(c_s) = \sum_{noun \in c_s} \frac{f(noun)}{|senses(noun)|} \tag{3.5}$$

To give a further example about the amount of information we were provided with after this process, the verb *swim* with the frame type `subj:pp.in` (indicating a subject and a prepositional phrase headed by *in*) had its strongest preferences for the WordNet class *fish* as subject and *body of water* as prepositional phrase object.
For this work, however, I restricted the possible conceptual classes within the frames' argument slots to 23 WordNet (mostly top) level nodes, to facilitate generalisation and comparison of the verbs' selectional preference behaviour, and defined abbreviations for them. Appendix 2 gives an overview of those WordNet synsets and its member nouns.

[16] For example, when considering the noun *coffee* isolated from its context, we do not know whether we are talking about the beverage *coffee*, the plant *coffee* or a *coffee* bean. Therefore, a third of the frequency of the noun was assigned to each of the three classes.

[17] Intuitively, Ribas' approach was an improvement to Resnik's in this detail, since Resnik split the number of times a certain noun appeared in an argument position by the total number of classes it appeared in, up to the top of the hierarchy. This treatment made the uncertainty dependent on the depth of the hierarchy, though, not from the number of different senses.

3.2.3 Clustering Verbs into Semantic Verb Classes

On the basis of the information about subcategorisation frame types and their arguments' conceptual classes I could start to cluster verbs. For that, I selected verbs from Levin's classification. The constraints I required for the verbs were (i) some verbs to be polysemous to investigate the realisation of the phenomenon by the clustering algorithms, and (ii) to distinguish between high and low frequent verbs to see the influence of the frequency onto the algorithms. Therefore I selected 153 different verbs with 226 verb senses which belonged to 30 different semantic classes. Four of the verbs were low-frequent verbs.

To cluster the verbs I applied two different algorithms, and each algorithm clustered the verbs both `(A)` according to only the syntactic information about the subcategorisation frames as acquired in section 3.2.1 and `(B)` according to the information about the subcategorisation frames including their selectional preferences as completed in section 3.2.2.

- *Iterative clustering based on a definition by Hughes (Hughes 1994):*

  In the beginning, each verb represented an own cluster. Iteratively, the distances between the clusters were measured and the closest clusters merged together.

  For the representation of the verbs, each verb v was assigned a distribution over the different types of subcategorisation frames t, according to the maximum likelihood estimate of `(A)` the verb appearing with the frame type:

  $$p(t|v) = \frac{f(v,t)}{f(v)} \tag{3.6}$$

  with $f(v,t)$ being the joint frequency of verb and frame type, and $f(v)$ being the frequency of the verb, both as determined in section 3.2.1,
  and `(B)` the verb appearing with the frame type and a selectionally preferred class combination C for the argument positions s in t:

  $$p(t, C|v) =_{def} p(t|v) * p(C|v,t) \tag{3.7}$$

  with $p(t|v)$ defined as in equation (3.6), and

  $$p(C|v,t) =_{def} \frac{\prod_{s \in t} ass(v_s, c_s)}{\sum_{c'_s \in class} \prod_{s \in t} ass(v_s, c'_s)} \tag{3.8}$$

  which intuitively estimates the probability of a certain class combination by comparing its association value with the sum over all possible class combinations, concerning the respective verb and frame.

  Starting out with each verb representing an own cluster, I iteratively determined the two closest clusters by applying the information-theoretic measure *relative entropy*[18] (Kull-

[18] Concerning the two typical problems one has with this measure, (i) zero frequencies were avoided by smoothing all frequencies by adding 0.5 to them, and (ii) since the measure is not symmetric, the respective smaller value was used as distance.

back and Leibler 1951) to compare the respective distributions. Those were merged into one cluster, and their distributions were merged by calculating a weighted average. Based on test runs I defined heuristics about how often the clustering was performed. In addition, I limited the maximum number of verbs within one cluster to four elements because otherwise the verbs showed the tendency to cluster together in a few large clusters only.

- *Unsupervised latent class analysis as described in Rooth (Rooth et al. 1998), based on the expectation-maximization algorithm:*

  The algorithm identified categorical types among indirectly observed multinomial distributions by applying the EM-algorithm (Dempster et al. 1977) to maximise the joint probability of `(A)` the verb and frame type: $p(v, t)$, and `(B)` the verb and frame type considering the selectional preferences: $p(v, t, C)$.

  It needed a fixed number of classes to be built and absolute frequencies of the verbs appearing with the subcategorisation frames. Test runs showed that 80 clusters modeled the semantic verb classes best. To be able to compare the analysis with the iterative clustering approach, I also limited the number of verbs within a cluster to four – considering that generally all verbs appear within each cluster when using this approach, the verbs with the highest respective probabilities where chosen.
  For version `(A)` the frequencies were provided by the joint frequencies of verbs and frame types, for version `(B)` I used the association values of the verbs with the frame types considering selectional preferences, as described by equation (3.7).

  The unsupervised algorithm then classified within 200 iterations joint events of verbs and subcategorisation frames into the 80 clusters τ, based on the iteratively estimated values

$$p(v, t) = \sum_{\tau} p(\tau, v, t) = \sum_{\tau} p(\tau) p(v|\tau) p(t|\tau) \tag{3.9}$$

$$p(v, t, C) = \sum_{\tau} p(\tau, v, t, C) = \sum_{\tau} p(\tau) p(v|\tau) p(t, C|\tau) \tag{3.10}$$

  for versions `(A)` and `(B)`, respectively.

3.3 Evaluation

The evaluation of the resulting clusters was adjusted to Levin's classification where the verbs had been taken from before. The following tables 3.1 and 3.2 present the success of the two clustering algorithms, considering the two different informational versions `(A)` and `(B)`. They contain the total number of clusters the algorithms had formed (all clusters containing between two and four verbs concerning the iterative algorithm, and a fixed number of 80 clusters

Information	Clusters		Verbs		Recall	Precision
	Total	Correct	Total	Correct		
SFs	31	20	90	55	36%	61%
SFs + Prefs	30	14	81	31	20%	38%

Figure 3.1: Iterative Clustering

Information	Clusters		Verbs(Senses)		Recall	Precision
	Total	Correct	Total	Correct		
SFs	80	36	107(159)	58(90)	38(40)%	54(57)%
SFs + Prefs	80	22	153(226)	47(56)	31(25)%	31(25)%

Figure 3.2: Latent Classes

concerning the latent class analysis), the share of correct clusters (those clusters which were subsets of a Levin class, for example the cluster containing the verbs *need, like, want, desire* is a subset of the Levin class *Desire*), and the number of verbs within those clusters. In table 3.2 the number of verbs in brackets refers to the respective number of their senses, since a verb could be clustered several times according to its senses, for example the verb *want* could be member of the classes *Desire* and *Declaration.*

Recall was defined by the percentage of verbs (verb senses) within the correct clusters compared to the total number of verbs (verb senses) to be clustered:

$$recall = \frac{|verbs_{correct\_clusters}|}{153} \quad (\frac{|verb\_senses_{correct\_clusters}|}{226}) \tag{3.11}$$

and precision was defined by the percentage of verbs (verb senses) appearing in the correct clusters compared to the number of verbs (verb senses) appearing in any cluster:

$$precision = \frac{|verbs_{correct\_clusters}|}{|verbs_{all\_clusters}|} \quad (\frac{|verb\_senses_{correct\_clusters}|}{|verb\_senses_{all\_clusters}|}) \tag{3.12}$$

Concerning precision, the assignment of verbs into semantic classes was most successful when using the iterative distance clustering method; 61% of all verbs were clustered into correct classes. Clustering the verbs into latent classes was with 54% comparably, but less successful. With both clustering methods the results became worse when adding information about the selectional preferences for the arguments in the subcategorisation frames.

3.4 Discussion

Following I present a choice of the correct clusters resulting from the different clustering approaches, in order to demonstrate that the classifications of both approaches illustrate the close relationship between the verbs' alternation behaviour and their affiliation to semantic classes: the resulting clusters which could be annotated by semantic class names show common alternation behaviour of their verbal elements.

The iteratively generated clusters show the verbs in the clusters followed by the five subcategorisation frame types with the highest probabilities in the overall verbs' distributions. The preferences for verbs in the *Desire* class were towards a subject followed by an infinitival phrase (`subj:to`). Alternatively a transitive `subj:obj` frame was used, partly followed by an additional infinitival phrase indicated by `to`:[19]

```
                need         * subj:to           0.382847629835582 *
                             * subj:obj0.318590601723132 *
                      * subj0.0962654034943192 *
       * subj:obj:to       0.0536333367658669 *
       * subj:obj:pp.for   0.0189647478804105 *

 like          * subj:to           0.344067278287462 *
               * subj:obj0.34302752293578 *
  * subj0.142110091743119 *
dv      0.0364220183486239 *
bj0.0262691131498471 *

 0.533195075557434 *
529642 *
```

lps to get

ved by an
e a piece
ably with

[19] It is striking that some wrong subcategorisation frames are listed, especially the intransitive frame type `subj`, which is partly due to underlying sentences containing an NP ellipsis (like in *"Our responsibilities are as follows: you invent, I commercialize."*), partly to parsing mistakes and the frame extraction.

```
roll        * subj(PhysObject)                  0.241451670685337 *
            * subj(PhysObject):adv              0.104624830989344 *
            * subj(Agent):obj(PhysObject)       0.0722786755339997 *
            * subj(LifeForm):obj(PhysObject)    0.0680756190652667 *
            * subj(Agent):obj(Part)             0.0525121359227189 *

fly         * subj(PhysObject)                  0.335013432064644 *
            * subj(PhysObject):adv              0.123622741498 *
            * subj(LifeForm):obj(PhysObject)    0.0657165877759204 *
            * subj(LifeForm):pp.to(LifeForm)    0.0452314211355251 *
            * subj(LifeForm):pp.to(Agent)       0.0438113663530466 *

move        * subj(PhysObject)                  0.200321615821647 *
            * subj(PhysObject):adv              0.11363088866625 *
            * subj(Part)                        0.0925972119246233 *
            * subj(Group):adv                   0.0442911091963341 *
            * subj(Part):adv                    0.0395279510615529 *
```

The latent class analysis resulted in clusters which are presented with their probability and the verbs with the highest probabilities for the respective cluster, according to cluster membership and combination with the subcategorisation frame types in the columns. The dot indicates whether the verb-frame combination was seen in the data.

Some verbs of *Telling* were clustered mainly according to their similar transitive use combined with an infinitival phrase:

Cluster PROB 0.0040		0.7455	0.0857	0.0482	0.0158
		subj:obj:to	subj	subj:obj	subj:pp.on
0.1734	advise	●	●	●	●
0.1213	teach	●	●	●	●
0.1198	instruct	●	●	●	●

The verbs of *Aspect* alternate between a subject only, realised by an activity, an inanimate subject followed by an infinitival phrase, and a living subject followed by a gerund:

Cluster PROB 0.0208		0.2203	0.1032	0.0942	0.0863
		subj(Action)	subj(PhysObject):to	subj(LifeForm):vger	subj(Agent):vger
0.3382	start	●	●	●	●
0.1945	finish	●		●	●
0.1846	stop	●		●	●
0.1584	begin	●	●		

Both approaches show that the relationship between alternation behaviour and semantic class could already be established when only considering information about the syntactic usage of the subcategorisation frames. The refinement by the frames' selectional preferences allowed further demarcations by the identification of conceptual restrictions on the use of the frames. Since the latent class analysis was able to assign verbs to several clusters, this further distinction can be referred to as distinguishing between the different verbs' senses and the respective uses of subcategorisation frames. For example, consider the following two clusters where the verb *play* was once clustered with *meet* because of the common strong tendency towards a transitive frame illustrating a general meeting, and once it was clustered with *fight* because of their common preference for an intransitive frame together with a prepositional phrase headed by *against*, when illustrating a more aggressive meeting like a match or a fight:

Cluster PROB 0.0095		0.5545	0.0468	0.0366	0.0340
		subj:obj	subj	subj:obj:pp.with	subj:obj:pp.at
0.4947	meet	●	●	●	●
0.1954	play	●	●	●	●

Cluster PROB 0.0018		0.1829	0.1297	0.0894	0.0693
		subj:pp.against	subj:obj	subj:obj:pp.against	subj:obj:adv
0.2212	fight	●	●	●	●
0.1959	play	●	●	●	●

An extensive investigation of the linguistic reliability of the verbs' and clusters' subcategorisation frames showed that the characterising usages could actually be underlined by example sentences, for example the above cited transitive use of the verb *fly* concerning the `subj:obj` frame type with a living subject and an inanimate object can be illustrated by the BNC-sentence *Today the older pilot flies the aircraft.*
This means that the linguistic properties as modelled for the approaches agree with (a selective part of) the verbs' properties. The clusters were therefore created on a reliable linguistic basis, an important fact to ensure, since an unreliable representation would question the successful relation between alternation behaviour and semantic classes.

A strange result seemed to be the fact that the clustering of the verbs became worse with both algorithms when taking the information about the frames' selectional preferences into account. This was due partly to the quality of the linguistic basis which has to be differentiated concerning the two informational versions: concerning version `(A)` there was little noise in the descriptions of the verbs' subcategorisation frames, as my study of linguistic reliability showed. Concerning version `(B)` the problems increased. Since the increase of noise correlated with the decrease of precision concerning the clustering success, this seemed an important factor to investigate: considering each argument slot within a subcategorisation frame on its own, the preferred conceptual classes illustrated linguistic reliable possibilities to insert arguments. But by the combination of the classes too many combinatorial possibilities had been created, so the combinations were not always possible to underlie with examples. The solution to this problem should be a different formulation of the conceptual class types, to ensure an improved token per type relation in order to avoid the data sparseness in tokens.

Both algorithms were confronted with two further problems:

- Polysemy:
  The different verb senses were hidden in the representation for one verb. That is, it was not obvious how to filter the uncertain number of senses out of the word-form. The iterative distance clustering completely failed to model verb senses; a polysemous verb was because of its opaque representation either not at all assigned to a cluster, or assigned to one cluster to which one of the verb's senses belongs. The latent class analysis was able to filter the multiple senses and assign them to distinct clusters, but tended to over-interpret.

- Low Frequency:
  Verbs which rarely appeared were difficult to cluster, since the necessary background was missing. The latent class analysis suffered from this sparse data, since those verbs were always assigned low probabilities. Distance clustering suffered even more, since – in addition to the sparse data concerning the verb's usage – also the information about the co-occurrence with subcategorisation frames was missing, so the verb's distribution contained mostly zeroes, a difficult mathematical basis.

Turning to the specific problems of the clustering algorithms, I first investigated the iterative clustering: letting each verb point to the closest verb as measured by relative entropy showed that 61%/36% in the respective versions chose a verb from the same semantic class. The conclusions from this investigation are two-fold: (i) the percentages can be considered as an upper boundary for what could have been achieved by the clustering method, since not more verbs than those pointing to a verb from the same class could be clustered correctly, so to achieve a better result other distance measures should be considered, and (ii) there is a loss of correct assignments when taking into account that – as table 3.1 shows – only 36%/20% of the verbs were finally found in correct clusters, which had to be caused by the merging process and the limit on the size of the clusters, so those were less than optimal and worth to be developed further.
Investigating the latent class analysis could underline that the data sparseness as mentioned before caused problems for the training process. In total there were only 6,873 verb-frame types for version (B) which was a too narrow basis. For version (A) I had 27,016 verb-frame types, but differently to (B) only 88 different frames, so creating 80 different clusters had the tendency to result in some classes where only one frame was favoured.

3.5 Conclusion

I proposed two algorithms for automatically classifying verbs semantically, based on their alternation behaviour. Taking Levin (Levin 1993) as golden standard for 153 manually chosen verbs with 226 verb senses and their assignment into 30 semantic classes, the iterative distance clustering succeeded for 61% of the verbs considering the syntactic usage of the frames only, and for 38% when adding information about the frames' arguments' selectional preferences. The latent class analysis succeeded for 54% and 31%, respectively.

An investigation of the resulting clusters showed that the assignment of the verbs was actually based on their shared linguistic properties: the verbs in a cluster presented a common alternation behaviour. The common properties within one cluster were refined when adding information about the selectional preferences to the syntactic description of the subcategorisation frames.

The discussion demonstrated that some problems in the classification process still have to be solved:

- An obvious problem in the clustering was the fact that the results were worse when incorporating the definition of the frames' selectional preferences. The representation of the subcategorisation frames including information about their selectional preferences should be improved to ensure a better token per type relation.
- The polysemy of verbs presented a problem, especially for the distance clustering, which could not distinguish between the multiple senses.

- Both approaches had difficulties in clustering low-frequency verbs, since the data could not be delimited in the clustering process.

Considering the overall motivation of this work, a successful step into the direction of presenting the connection between the verbs' alternation behaviour and their semantics by automatic means is done. Naturally, there are possibilities to improve the process.

Appendix 1: Subcategorisation Frames

Part 1.1 contains a list of the 88 subcategorisation frame types which built the basis for the syntactic description of the verbs. The frames are numbered from 0 to 87. Explanations about the syntactic features within the frames can be found in part 1.2. The appendix is concluded in part 1.3 by the joint frequencies of the verb *give* concerning the frame types.

1.1 Frame Types

0	subj
1	subj:adv
2	subj:ap
3	subj:obj
4	subj:obj:adv
5	subj:obj:ap
6	subj:obj:as
7	subj:obj:obj
8	subj:obj:obj:adv
9	subj:obj:obj:pp.at
10	subj:obj:obj:pp.for
11	subj:obj:obj:pp.in
12	subj:obj:obj:pp.on
13	subj:obj:obj:pp.to
14	subj:obj:obj:pp.with
15	subj:obj:pp.about
16	subj:obj:pp.after
17	subj:obj:pp.against
18	subj:obj:pp.as
19	subj:obj:pp.at
20	subj:obj:pp.before
21	subj:obj:pp.between
22	subj:obj:pp.by
23	subj:obj:pp.during
24	subj:obj:pp.for
25	subj:obj:pp.from
26	subj:obj:pp.in
27	subj:obj:pp.in:adv
28	subj:obj:pp.in:pp.in
29	subj:obj:pp.into
30	subj:obj:pp.like
31	subj:obj:pp.of
32	subj:obj:pp.on
33	subj:obj:pp.out_of
34	subj:obj:pp.over
35	subj:obj:pp.through
36	subj:obj:pp.to

37	subj:obj:pp.under
38	subj:obj:pp.with
39	subj:obj:pp.within
40	subj:obj:pp.without
41	subj:obj:ppart
42	subj:obj:s
43	subj:obj:sub
44	subj:obj:that
45	subj:obj:to
46	subj:obj:vbase
47	subj:obj:vger
48	subj:pp.about
49	subj:pp.across
50	subj:pp.after
51	subj:pp.against
52	subj:pp.as
53	subj:pp.at
54	subj:pp.at:adv
55	subj:pp.between
56	subj:pp.by
57	subj:pp.for
58	subj:pp.for:adv
59	subj:pp.from
60	subj:pp.from:pp.to
61	subj:pp.in
62	subj:pp.in:adv
63	subj:pp.into
64	subj:pp.like
65	subj:pp.of
66	subj:pp.on
67	subj:pp.on:adv
68	subj:pp.out_of
69	subj:pp.over
70	subj:pp.through
71	subj:pp.to
72	subj:pp.to:adv
73	subj:pp.towards
74	subj:pp.under
75	subj:pp.up_to
76	subj:pp.upon
77	subj:pp.with
78	subj:pp.with:adv
79	subj:ppart
80	subj:s
81	subj:sub
82	subj:that
83	subj:to
84	subj:to:adv

85	subj:vbase
86	subj:vbase:adv
87	subj:vger

1.2 Frame Features

Syntactic features of the frame types, as defined by the English grammar:

adv	adverb
ap	adjectival phrase
as	as-expression
pp	prepositional phrase
ppart	stranded preposition
s	sentence
that	subordinated that-phrase
to	infinitive form of verb after 'to'
vbase	base form of verb
vger	gerund

and additional identifiers:

subj	subject of the sentence
obj	object of the sentence

1.3 Joint Frequencies of the Verb *give* concerning the Frame Types

The following list displays the joint frequencies of the verb *give* concerning the frame types in column two. For frame types defined in appendix 3.5 which do not appear here the joint frequency was zero.

give	subj	758
give	subj:adv	105
give	subj:ap	58
give	subj:obj	9,982
give	subj:obj:adv	498
give	subj:obj:ap	60
give	subj:obj:as	53
give	subj:obj:obj	13,430
give	subj:obj:obj:adv	158
give	subj:obj:obj:pp.at	59
give	subj:obj:obj:pp.for	144
give	subj:obj:obj:pp.in	238
give	subj:obj:obj:pp.on	68
give	subj:obj:obj:pp.to	240
give	subj:obj:obj:pp.with	39
give	subj:obj:pp.about	57
give	subj:obj:pp.after	42
give	subj:obj:pp.against	14
give	subj:obj:pp.as	171

give	subj:obj:pp.at	220
give	subj:obj:pp.before	24
give	subj:obj:pp.between	5
give	subj:obj:pp.by	40
give	subj:obj:pp.during	30
give	subj:obj:pp.for	566
give	subj:obj:pp.from	56
give	subj:obj:pp.in	936
give	subj:obj:pp.in:adv	16
give	subj:obj:pp.in:pp.in	11
give	subj:obj:pp.into	17
give	subj:obj:pp.like	8
give	subj:obj:pp.of	198
give	subj:obj:pp.on	234
give	subj:obj:pp.out_of	16
give	subj:obj:pp.over	35
give	subj:obj:pp.through	15
give	subj:obj:pp.to	3,735
give	subj:obj:pp.under	26
give	subj:obj:pp.with	103
give	subj:obj:pp.within	15
give	subj:obj:pp.without	36
give	subj:obj:ppart	98
give	subj:obj:s	35
give	subj:obj:sub	16
give	subj:obj:that	67
give	subj:obj:to	277
give	subj:obj:vbase	15
give	subj:obj:vger	35
give	subj:pp.about	4
give	subj:pp.across	3
give	subj:pp.after	5
give	subj:pp.against	1
give	subj:pp.as	10
give	subj:pp.at	17
give	subj:pp.at:adv	3
give	subj:pp.between	1
give	subj:pp.by	2
give	subj:pp.for	34
give	subj:pp.for:adv	6
give	subj:pp.from	5
give	subj:pp.from:pp.to	1
give	subj:pp.in	50
give	subj:pp.into	9
give	subj:pp.of	31
give	subj:pp.on	14
give	subj:pp.out_of	6
give	subj:pp.over	1

give	subj:pp.through	2
give	subj:pp.to	288
give	subj:pp.to:adv	17
give	subj:pp.towards	2
give	subj:pp.under	3
give	subj:pp.up_to	6
give	subj:pp.upon	3
give	subj:pp.with	14
give	subj:pp.with:adv	1
give	subj:ppart	6
give	subj:s	280
give	subj:sub	1
give	subj:that	18
give	subj:to	38
give	subj:vbase	36
give	subj:vbase:adv	1
give	subj:vger	15

Appendix 2: WordNet (Top) Synsets

There are 11 top level nodes of 11 hierarchies in WordNet. Since the concept of `Entity` seemed too general as conceptual class, I replaced it by the next lower levels (13 different synsets). Each WordNet synset is defined by an identifying abbreviation, followed by the nouns which are member of that synset:

```
Entity:      entity
             => LifeForm:      life form, organism, being, living thing
             => Cell:          cell
             => Agent:         causal agent, cause, causal agency
             => PhysObject:    object, inanimate object, physical object
             => Thing:         thing
             => Whole:         whole, whole thing, unit
             => Content:       subject, content, depicted object
             => Unit:          unit, building block
             => Part:          part, piece
             => Essential:     necessity, essential, requirement,
                               requisite, necessary, need
             => Inessential:   inessential
             => Variable:      variable
             => Anticipation:  anticipation
Psycho:      psychological_feature
Abstract:    abstraction
Location:    location
Shape:       shape, form
State:       state
Event:       event
Action:      act, human_action, human_activity
Group:       group, grouping
Possession:  possession
Phenomenon:  phenomenon
```

Inside-Outside Estimation of a Lexicalized PCFG for German

– Gold –

Franz Beil, Glenn Carroll, Detlef Prescher, Stefan Riezler, and Mats Rooth

Abstract

The paper describes an extensive experiment in inside-outside estimation of a lexicalized probabilistic context free grammar for German verb-final clauses. Grammar and formalism features which make the experiment feasible are described. Successive models are evaluated on precision and recall of phrase markup.

4.1 Introduction

Charniak (1995) and Carroll and Rooth (1998) present head-lexicalized probabilistic context free grammar formalisms, and show that they can effectively be applied in inside-outside estimation of syntactic language models for English, the parameterization of which encodes lexicalized rule probabilities and syntactically conditioned word-word bigram collocates. The present paper describes an experiment where a slightly modified version of Carroll and Rooth's model was applied in a systematic experiment on German, which is a language with rich inflectional morphology and free word order (or rather, compared to English, free-er phrase order). We emphasize techniques which made it practical to apply inside-outside estimation of a lexicalized context free grammar to such a language. These techniques relate to the treatment of argument cancellation and scrambled phrase order; to the treatment of case features in category labels; to the category vocabulary for nouns, articles, adjectives and their projections; to lexicalization based on uninflected lemmata rather than word forms; and to exploitation of a parameter-tying feature.

4.2 Corpus and morphology

The data for the experiment is a corpus of German subordinate clauses extracted by regular expression matching from a 200 million token newspaper corpus. The clause length ranges between four and 12 words. Apart from infinitival VPs as verbal arguments, there are no further clausal embeddings, and the clauses do not contain any punctuation except for a terminal period. The corpus contains 4128873 tokens and 450526 clauses which yields an average of 9.16456 tokens per clause. Tokens are automatically annotated with a list of part-of-speech (PoS) tags using a computational morphological analyser based on finite-state technology (Karttunen et al. (1994), Schiller and Stöckert (1995)).

A problem for practical inside-outside estimation of an inflectional language like German arises with the large number of terminal and low-level non-terminal categories in the grammar resulting from the morpho-syntactic features of words. Apart from major class (noun, adjective, and so forth) the analyser provides an ambiguous word with a list of possible combinations of inflectional features like gender, person, number (cf. the top part of Fig. 4.1 for an example ambiguous between nominal and adjectival PoS; the PoS is indicated following the '+' sign; the features enclosed between '^' and '+' indicate the derivation if available; the entry is headed by the lemma of the analysandum; `Pos` is the feature for an adjective's positive form as opposed to comparative or superlative; the '*'-sign marks uppercase forms.). In order to reduce the number of parameters to be estimated, and to reduce the size of the parse forest used in inside-outside estimation, we collapsed the inflectional readings of adjectives, adjective derived nouns, article words, and pronouns to a single morphological feature (see separated last line in Fig.

```
analyze> Deutsche
 1. deutsch^ADJ.Pos+NN.Fem.Akk.Sg
 2. deutsch^ADJ.Pos+NN.Fem.Nom.Sg
 3. deutsch^ADJ.Pos+NN.Masc.Nom.Sg.Sw
 4. deutsch^ADJ.Pos+NN.Neut.Akk.Sg.Sw
 5. deutsch^ADJ.Pos+NN.Neut.Nom.Sg.Sw
 6. deutsch^ADJ.Pos+NN.NoGend.Akk.Pl.St
 7. deutsch^ADJ.Pos+NN.NoGend.Nom.Pl.St
 8. *deutsch+ADJ.Pos.Fem.Akk.Sg
 9. *deutsch+ADJ.Pos.Fem.Nom.Sg
 10. *deutsch+ADJ.Pos.Masc.Nom.Sg.Sw
 1. *deutsch+ADJ.Pos.Neut.Akk.Sg.Sw
12. *deutsch+ADJ.Pos.Neut.Nom.Sg.Sw
13. *deutsch+ADJ.Pos.NoGend.Akk.Pl.St
14. *deutsch+ADJ.Pos.NoGend.Nom.Pl.St

==>  Deutsche  { ADJ.E, NNADJ.E }
```

Figure 4.1: Collapsing Inflectional Features

```
während   { ADJ.Adv, ADJ.Pred, KOUS, APPR.Dat, APPR.Gen }
sich  { PRF.Z }
das   { DEMS.Z, ART.Def.Z }
Preisniveau   { NN.Neut.NotGen.Sg }
dem   { DEMS.M, ART.Def.M }
westdeutschen     { ADJ.N }
annähere  { VVFIN }
. { PER }
```

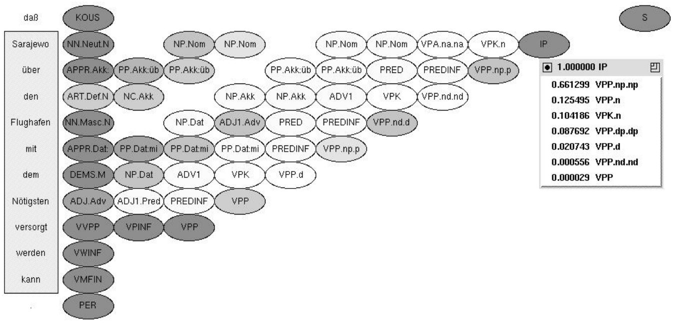
Figure 4.2: Corpus Clip

4.1 for an example). This reduced the number of low-level categories, as exemplified in Fig. 4.2: *das* has one reading as an article and one as a demonstrative; *westdeutschen* has one reading as an adjective, with its morphological feature N indicating the inflectional suffix.

We use the special tag `UNTAGGED` indicating that the analyser fails to provide a tag for the word. The vast majority of `UNTAGGED` words are proper names not recognized as such. These gaps in the morphology have little effect on our experiment.

4.3 Grammar

The grammar is a manually developed headed context-free phrase structure grammar for German subordinate clauses with 5508 rules and 562 categories, 209 of which are terminal categories. The formalism is that of Carroll and Rooth (1998), henceforth C+R:


daß
Sarajewo
über
den
Flughafen
mit
dem
Nötigsten
versorgt
werden
kann
KOUS
S
NN.Neut.N NP.Nom NP.Nom NP.Nom NP.Nom VPA.na.na VPK.n IP
APPR.Akk: PP.Akk:üb PP.Akk:üb PP.Akk:üb PP.Akk:üb PRED PREDINF VPP.np.p
ART.Def.N NC.Akk NP.Akk NP.Akk ADV1 VPK VPP.nd.nd
NN.Masc.N NP.Dat ADJ1.Adv PRED PREDINF VPP.nd.d
APPR.Dat: PP.Dat:mi PP.Dat:mi PP.Dat:mi PREDINF VPP.np.p
DEMS.M NP.Dat ADV1 VPK VPP.d
ADJ.Adv ADJ1.Pred PREDINF VPP
VVPP VPINF VPP
VWINF
VMFIN
PER

```
● 1.000000 IP
  0.661299 VPP.np.np
  0.125495 VPP.n
  0.104186 VPK.n
  0.087692 VPP.dp.dp
  0.020743 VPP.d
  0.000556 VPP.nd.nd
  0.000029 VPP
```

Word-by-word gloss of the clause on the left:
'that Sarajevo over the airport with the essentials supplied will can'

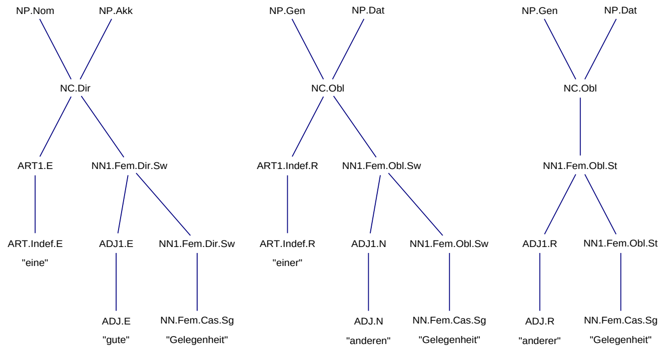
Figure 4.3: Chart browser

```
mother -> non-heads head' non-heads (freq)
```

The rules are head marked with a prime. The non-head sequences may be empty. `freq` is a rule frequency, which is initialized randomly and subsequently estimated by the inside outside-algorithm. To handle systematic patterns related to features, rules were generated by Lisp functions, rather than being written directly in the above form. With very few exceptions (rules for coordination, S-rule), the rules do not have more than two daughters.

Grammar development is facilitated by a chart browser that permits a quick and efficient discovery of grammar bugs (Carroll 1997a). Fig. 4.3 shows that the ambiguity in the chart is quite considerable even though grammar and corpus are restricted. For the entire corpus, we computed an average 9202 trees per clause. In the chart browser, the categories filling the cells indicate the most probable category for that span with their estimated frequencies. The pop-up window under `IP` presents the ranked list of all possible categories for the covered span. Rules (chart edges) with frequencies can be viewed with a further menu. In the chart browser, colors are used to display frequencies (between 0 and 1) estimated by the inside-outside algorithm. This allows properties shared across tree analyses to be checked at a glance; often grammar and estimation bugs can be detected without mouse operations.

The grammar covers 88.5% of the clauses and 87.9% of the tokens contained in the corpus. Parsing failures are mainly due to `UNTAGGED` words contained in 6.6% of the failed clauses, the pollution of the corpus by infinitival constructions (≈1.3%), and a number of coordinations not covered by the grammar (≈1.6%).



Glosses:*'a good opportunity'*, *'a different opportunity'*, *'different opportunity'*

Figure 4.4: Noun Projections

4.3.1 Case features and agreement

On nominal categories, in addition to the four cases `Nom`, `Gen`, `Dat`, and `Akk`, case features with a disjunctive interpretation (such as `Dir` for `Nom` or `Akk`) are used. The grammar is written in such a way that non-disjunctive features are introduced high up in the tree. Figure 4.4 illustrates the use of disjunctive features in noun projections: The terminal `NN` contains the four-way ambiguous `Cas` case feature; the N-bar (`NN1`) and noun chunk `NC` projections disambiguate to two-way ambiguous case features `Dir` and `Obl`; the weak/strong (`Sw/St`) feature of `NN1` facilitates or prevents combination with a determiner, respectively; only as soon as the `NP` projection, the case feature appears in disambiguated form. The use of disjunctive case features results in some reduction in the size of the parse forest, and some parameter pooling. Essentially the full range of agreement inside the noun phrase is enforced. Agreement between the nominative NP and the tensed verb (e.g. in number) is not enforced by the grammar, in order to control the number of parameters and rules.

For noun phrases we employ Abney's chunk grammar organization (Abney 1996). The noun chunk (`NC`) is an approximately non-recursive projection that excludes post-head complements and (adverbial) adjuncts introduced higher than pre-head modifiers and determiners but includes participial pre-modifiers with their complements. Since we perform complete context free parsing, parse forest construction, and inside-outside estimation, chunks are not motivated by deterministic parsing. Rather, they facilitate evaluation and graphical debugging, by tending to increase the span of constituents with high estimated frequency.

class	#	frame types
VPA	15	n, na, nad, nai, nap, nar, nd, ndi, ndp, ndr, ni, nir, np, npr, nr
VPP	13	d, di, dp, dr, i, ir, n, nd, ni, np, p, pr, r
VPI	10	a, ad, ap, ar, d, dp, dr, p, pr, r
VPK	2	i, n

Table 4.1: Number and types of verb frames

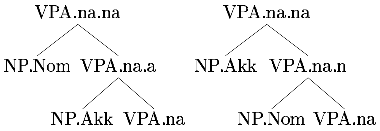


Figure 4.5: Coding of canonical and scrambled argument order

4.3.2 Subcategorisation frames of verbs

The grammar distinguishes four subcategorisation frame classes: active (VPA), passive (VPP), infinitival (VPI) frames, and copula constructions (VPK). A frame may have maximally three arguments. Possible arguments in the frames are nominative (n), dative (d) and accusative (a) NPs, reflexive pronouns (r), PPs (p), and infinitival VPs (i). The grammar does not distinguish plain infinitival VPs from *zu*-infinitival VPs. The grammar is designed to partially distinguish different PP frames relative to the prepositional head of the PP. A distinct category for the specific preposition becomes visible only when a subcategorized preposition is cancelled from the subcat list. This means that specific prepositions do not figure in the evaluation discussed below. The number and the types of frames in the different frame classes are given in Table 4.1.

German, being a language with comparatively free phrase order, allows for scrambling of arguments. Scrambling is reflected in the particular sequence in which the arguments of the verb frame are saturated. Compare Figure 4.5 for an example of a canonical subject-object order in an active transitive frame and its scrambled object-subject order. The possibility of scrambling verb arguments yields a substantial increase in the number of rules in the grammar (e.g. 102 combinatorically possible argument rules for all in VPA frames). Adverbs and non-subcategorized PPs are introduced as adjuncts to VP categories which do not saturate positions in the subcat frame.

In earlier experiments, we employed a flat clausal structure, with rules for all permutations of complements. As the number of frames increased, this produced prohibitively many rules, particularly with the inclusion of adjuncts.

4.4 Parameters

The parameterization is as in C+R, with one significant modification. Parameters consist of (i) rule parameters, corresponding to right hand sides conditioned by parent category and parent head; (ii) lexical choice parameters for non-head children, corresponding to child lemma conditioned by child category, parent category, and parent head lemma. See C+R or Charniak (1995) for an explanation of how such parameters define a probabilistic weighting of trees. The change relative to C+R is that lexicalization is by uninflected lemma rather than word form. This reduces the number of lexical parameters, giving more acceptable model sizes and eliminating splitting of estimated frequencies among inflectional forms. Inflected forms are generated at the leaves of the tree, conditioned on terminal category and lemma. This results in a third family of parameters, though usually the choice of inflected form is deterministic.

A parameter pooling feature is used for argument filling where all parent categories of the form VP.x.y are mapped to a category VP.x in defining lexical choice parameters. The consequence is e.g. that an accusative daughter of a nominative-accusative verb uses the same lexical choice parameter, whether a default or scrambled word order is used. (This feature was used by C+R for their phrase trigram grammar, not in the linguistic part of their grammar.) Not all desirable parameter pooling can be expressed in this way, though; for instance rule parameters are not pooled, and so get split when the parent category bears an inflectional feature.

4.5 Estimation

The training of our probabilistic CFG proceeds in three steps: (i) unlexicalized training with the `supar` parser, (ii) bootstrapping a lexicalized model from the trained unlexicalized one with the `ultra` parser, and finally (iii) lexicalized training with the `hypar` parser (Carroll 1997b). Each of the three parsers uses the inside-outside algorithm. `supar` and `ultra` use an unlexicalized weighting of trees, while `hypar` uses a lexicalized weighting of trees. `ultra` and `hypar` both collect frequencies for lexicalized rule and lexical choice events, while `supar` collects only unlexicalized rule frequencies.

Our experiments have shown that training an unlexicalized model first is worth the effort. Despite our use of a manually developed grammar that does not have to be pruned of superfluous rules like an automatically generated grammar, the lexicalized model is notably better when preceded by unlexicalized training (see also Ersan and Charniak (1995) for related observations). A comparison of immediate lexicalized training (without prior training of an unlexicalized model) and our standard training regime that involves preliminary unlexicalized training speaks in favor of our strategy (cf. the different 'lex 0' and 'lex 2' curves in figures 4.7 and 4.8). However, the amount of unlexicalized training has to be controlled in some way.

A		**B**		**C**	
1:	52.0199	1:	53.7654	1:	49.8165
2:	25.3652	2:	26.3184	2:	23.1008
3:	24.5905	3:	25.5035	3:	22.4479
⋮	⋮	⋮	⋮	⋮	⋮
13:	24.2872	55:	25.0548	70:	22.1445
14:	24.2863	56:	25.0549	80:	22.1443
15:	24.2861	57:	25.0549	90:	22.1443
16:	24.2861	58:	25.0549	95:	22.1443
17:	24.2867	59:	25.055	96:	22.1444

Table 4.2: Overtraining (iteration: cross-entropy on heldout data)

A standard criterion to measure overtraining is to compare log-likelihood values on held-out data of subsequent iterations. While the log-likelihood value of the training data is theoretically guaranteed to converge through subsequent iterations, a decreasing log-likelihood value of the held-out data indicates overtraining. Instead of log-likelihood, we use the inversely proportional cross-entropy measure. Table 4.2 shows comparisons of different sizes of training and heldout data (training/heldout): (A) 50k/50k, (B) 500k/500k, (C) 4.1M/500k. The overtraining effect is indicated by the increase in cross-entropy from the penultimate to the ultimate iteration in the tables. Overtraining results for lexicalized models are not yet available.

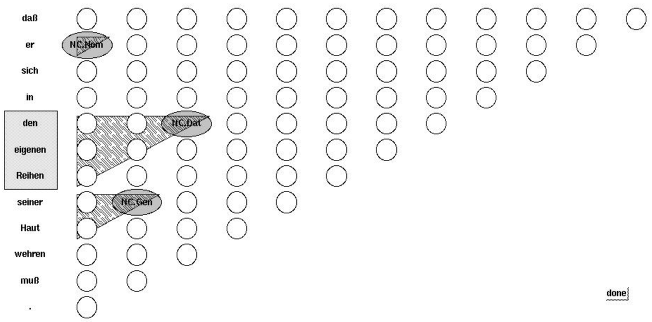
However, a comparison of precision/recall measures on categories of different complexity through iterative unlexicalized training shows that the mathematical criterion for overtraining may lead to bad results from a linguistic point of view. While we observed more or less converging precision/recall measures for lower level structures such as noun chunks, iterative unlexicalized training up to the overtraining threshold turned out to be disastrous for the evaluation of complex categories that depend on almost the entire span of the clause. The recognition of subcategorization frames through 60 iterations of unlexicalized training shows a massive decrease in precision/recall from the best to the last iteration, even dropping below the results with the randomly initialized grammar (see Figure 4.8).

4.5.1 Training regime

We compared lexicalized training with respect to different starting points: a random unlexicalized model, the trained unlexicalized model with the best precision/recall results, and an unlexicalized model that comes close to the cross-entropy overtraining threshold. (For the effect of differently randomized rule frequencies, we refer the reader to Appendix 2.) The details of the training steps are as follows:

(1) 0, 2 and 60 iterations of unlexicalized parsing with `supar`;

(2) lexicalization with `ultra` using the entire corpus;



Word-by-word gloss of the clause:
'that he himself in his own ranks hiw skin defend must'

Figure 4.6: Chart browser for manual NC labelling

(3) 23 iterations of lexicalized parsing with `hypar`.

The training was done simultaneously on four machines (two 167 MHz UltraSPARC with 188 MB and 312 MB main memory, and two 296 MHz SUNW UltraSPARC-II with 1.1 GB main memory). Using the grammar described here, one iteration of `supar` on the entire corpus takes about 2.5 hours, lexicalization and generating an initial lexicalized model takes more than six hours, and an iteration of lexicalized parsing can be done in 5.5 hours.

4.6 Evaluation

For the evaluation, a total of 600 randomly selected clauses were manually annotated by two labellers. Using a chart browser, the labellers filled the appropriate cells with category names `NC` and `PPART`, and those of maximal VP projections (cf. Figure 4.6 for an example of NC-labelling). We included prepositional phrases with preposition incorporated determiners (i.e. `PPART`, e.g. *'beim Zahnarzt'*, *'at the dentist'*) in the set of annotated noun chunks because the left boundary of the `NC` contained within the prepositional phrase is incorporated into the preposition word. Subsequent alignment of the labellers decisions resulted in a total of 1353 labelled NC categories: 627 `NC.Nom`, 319 `NC.Akk`, 253 `NC.Dat`, 75 `NC.Gen`, 73 `PPART.Dat`, and 6 `PPART.Akk`. The total of 584 labelled VP categories subdivides into 21 different verb frames with 340 different lemma heads. The dominant frames are active transitive (164 occurrences) and active intransitive (117 occurrences). They represent almost half of the annotated frames. Thirteen frames occur less than ten times, five of which just once (compare Table 4.3 for the numbers of individual frames).

VPA.na	164	VPP	4
VPA.n	117	VPP.p	3
VPK.n	90	VPA.npr	3
VPP.n	64	VPA.ni	2
VPA.np	62	VPA.ndr	2
VPA.nr	19	VPP.d	1
VPA.nd	16	VPI.p	1
VPA.nap	12	VPI.a	1
VPP.np	9	VPA.nir	1
VPA.nad	8	VPA.ndp	1
VPP.nd	4		

Table 4.3: Frames in the test set

4.6.1 Methodology

To evaluate iterative training, we extracted maximum probability (Viterbi) trees for the 600 clause test set in each iteration of parsing. For extraction of a maximal probability parse in unlexicalized training, we used Schmid's `lopar` parser (Schmid 1999). Trees were mapped to a database of parser generated markup guesses, and we measured *precision* and *recall* against the manually annotated category names and spans. Precision gives the ratio of correct guesses over all guesses, and recall the ratio of correct guesses over the number of phrases identified by human annotators. Here, we render only the precision/recall results on pairs of category names and spans, neglecting less interesting measures on spans alone. For the figures of adjusted recall, the number of unparsed misses has been subtracted from the number of possibilities.

In the following, we focus on the combination of the best unlexicalized model and the lexicalized model that is grounded on the former. In addition, as mentioned in section 4.5.1, we compare the results for our best lexicalized model (i) to a trained lexicalized model resulting from lexicalization of a random grammar and (ii) to a trained lexicalized model derived from an excessively trained unlexicalized model. The precision and recall plots for those models are labelled `lex 00` and `lex 60`, respectively.

Furthermore, in Appendix 3, we compare the training results with respect to varying initial random states of the grammar.

4.6.2 NC Evaluation

Figure 4.7 plots precision/recall for the training runs described in section 4.5.1, with lexicalized parsing starting after 0, 2, or 60 unlexicalized iterations. The best results are achieved by starting with lexicalized training after two iterations of unlexicalized training. Of a total of 1353 annotated `NC`s with case, 1103 are correctly recognized in the best unlexicalized model and 1112 in the last lexicalized model. With a number of 1295 guesses in the unlexicalized and 1288 guesses in the final lexicalized model, we gain 1.2% in precision (85.1% vs. 86.3%) and 0.6% in recall (81.5% vs. 82.1%) through lexicalized training. Adjustment to parsed clauses yields 88% vs. 89.2% in recal. As shown in Figure 4.7, the gain is achieved already within the first iteration; it is equally distributed between corrections of category boundaries and labels.

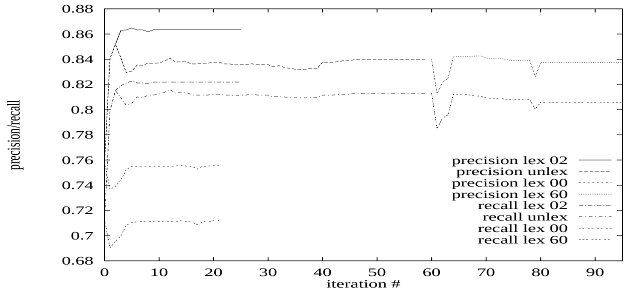


Figure 4.7: Precision/recall measures on `NC`s with case (random 0)

The comparatively small gain with lexicalized training could be viewed as evidence that the chunking task is too simple for lexical information to make a difference. However, we find about 7% revised guesses from the unlexicalized to the first lexicalized model. Currently, we do not have a clear picture of the newly introduced errors.

The plots labeled "00" are results for lexicalized training starting from a random initial grammar. The precision measure of the first lexicalized model falls below that of the unlexicalized random model (74%), only recovering through lexicalized training to equalize the precision measure of the random model (75.6%). This indicates that some degree of unlexicalized initialization is necessary, if a good lexicalized model is to be obtained.

Skut and Brants (1998) report 84.4% recall and 84.2% for NP and PP chunking without case labels. While these are numbers for a simpler problem and are slightly below ours, they are figures for an experiment on unrestricted sentences. A genuine comparison has to await extension of our model to free text.

4.6.3 Verb Frame Evaluation

Figure 4.8 gives results for verb frame recognition under the same training conditions. Again, we achieve best results by lexicalizing the second unlexicalized model. Of a total of 584 annotated verb frames, 384 are correctly recognized in the best unlexicalized model and 397 through subsequent lexicalized training. Precision for the best unlexicalized model is 68.4%. This is raised by 2% to 70.4% through lexicalized training; recall is 65.7%/68%; adjustment by 41 unparsed misses makes for 70.4%/72.8% in recal. The rather small improvements are in contrast to 88 differences in parser markup, i.e. 15.7%, between the unlexicalized and second lexicalized model. The main gain is observed within the first two iterations (cf. Figure 4.8; for readability, we dropped the recall curves when more or less parallel to the precision curves).

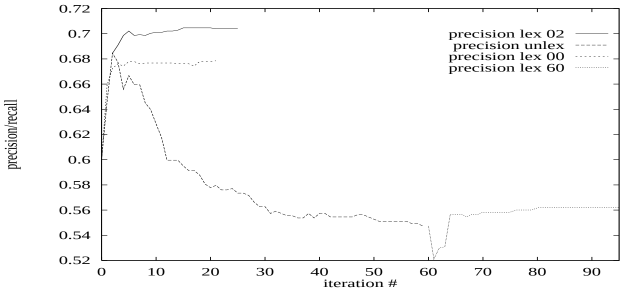


Figure 4.8: Precision measures on all verb frames (random 0)

Results for lexicalized training without prior unlexicalized training are better than in the NC evaluation, but fall short of our best results by more than 2%.

The most notable observation in verb frame evaluation is the decrease of precision of frame recognition in unlexicalized training from the second iteration onward. After several dozen iterations, results are 5% below a random model and 14% below the best model. The primary reason for the decrease is the mistaken revision of adjoined PPs to argument PPs. E.g. the required number of 164 transitive frames is missed by 76, while the parser guesses 64 `VPA.nap` frames in the final iteration against the annotator's baseline of 12. In contrast, lexicalized training generally stabilizes w.r.t. frame recognition results after only few iterations.

The plot labeled "lex 60" gives precision for a lexicalized training starting from the unlexicalized model obtained with 60 iterations, which measured by linguistic criteria is a very poor state. As far as we know, lexicalized EM estimation never recovers from this bad state.

4.6.4 Evaluation of non-PP Frames

Because examination of individual cases showed that PP attachments are responsible for many errors, we did a separate evaluation of non-PP frames. We filtered out all frames labelled with a PP argument from both the maximal probability parses and the manually annotated frames (91 filtered frames), measuring precision and recall against the remaining 493 labeller annotated non-PP frames.

For the best lexicalized model, we find somewhat but not excessively better results than those of the evaluation of the entire set of frames. Of 527 guessed frames in parser markup, 382 are correct, i.e. a precision of 72.5%. The recall figure of 77.5% is considerably better since

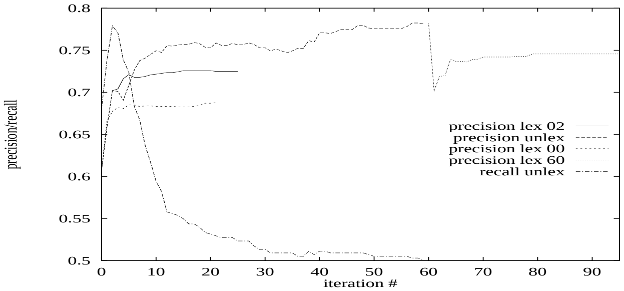


Figure 4.9: Precision measures on non-PP frames (random 0)

overgeneration of 34 guesses is neglected. The differences with respect to different starting points for lexicalization emulate those in the evaluation of all frames.

The rather spectacular looking precision and recall differences in unlexicalized training confirm what was observed for the full frame set. From the first trained unlexicalized model throughout unlexicalized training, we find a steady increase in precision (70% first trained model to 78% final model) against a sharp drop in recall (78% peek in the second model vs. 50% in the final). Considering our above remarks on the difficulties of frame recognition in unlexicalized training, the sharp drop in recall is to be expected: Since recall measures the correct parser guesses against the annotator's baseline, the tendency to favor PP-arguments over PP-adjuncts leads to a loss in guesses when PP-frames are abandoned. Similarly, the rise in precision is mainly explained by the decreasing number of guesses when cutting out non-PP frames. For further discussion of what happens with individual frames, see Appendix 1.

One systematic result in these plots is that performance of lexicalized training stabilizes after a few iterations. This is consistent with what happens with rule parameters for individual verbs, which are close to their final values within five iterations.

4.7 Conclusion

Our principal result is that scrambling-style free-er phrase order, case morphology and subcategorization, and NP-internal gender, number and case agreement can be dealt with in a head-lexicalized PCFG formalism by means of carefully designed categories and rules which limit the size of the packed parse forest and give desirable pooling of parameters. Hedging this, we point out that we made compromises in the grammar (notably, in not enforcing nominative-verb agreement) in order to control the number of categories, rules, and parameters.

A second result is that iterative lexicalized inside-outside estimation appears to be beneficial, although the precision/recall increments are smal. We believe this is the first substantial investigation of the utility of iterative lexicalized inside-outside estimation of a lexicalized probabilistic grammar involving a carefully built grammar where parses can be evaluated by linguistic criteria.

A third result is that using too many unlexicalized iterations (more than two) is detrimental. A criterion using cross-entropy overtraining on held-out data dictates many more unlexicalized iterations, and this criterion is therefore inappropriate.

Finally, we have clear cases of lexicalized EM estimation being stuck in linguistically bad states. As far as we know, the model which gave the best results could also be stuck in a comparatively bad state. We plan to experiment with other lexicalized training regimes, such as ones which alternate between different training corpora.

The experiments are made possible by improvements in parser and hardware speeds, the carefully built grammar, and evaluation tools. In combination, these provide a unique environment for investigating training regimes for lexicalized PCFGs. Much work remains to be done in this area, and we feel that we are just beginning to develop understanding of the time course of parameter estimation, and of the general efficacy of EM estimation of lexicalized PCFGs as evaluated by linguistic criteria.

We believe our current grammar of German could be extended to a robust free-text chunk/phrase grammar in the style of the English grammar of Carroll and Rooth (1998) with about a month's work, and to a free-text grammar treating verb-second clauses and additional complementation structures (notably extraposed clausal complements) with about one year of additional grammar development and experiment. These increments in the grammar could easily double the number of rules. However this would probably not pose a problem for the parsing and estimation software.

Appendix 1: Evaluation of Individual Frames

	all guesses / correct guesses					
iter.	VPA.n	VPA.na	VPA.nd	VPA.nr	VPA.np	VPA.nap
possible	**117**	**164**	**16**	**19**	**62**	**12**
random	213/ 96	107/ 94	35/ 6	34/ 19	0/ 0	1/ 0
unlex 2	152/ 86	161/ 131	19/ 7	31/ 19	2/ 0	0/ 0
unlex 60	53 / 46	89/ 79	7/ 5	11/ 8	92/ 40	63/ 7
unlex 100	53 / 46	83/ 74	7/ 5	11/ 8	90/ 40	65/ 8
lex-00 1	196/ 99	129/ 117	28/ 7	34/ 19	0/ 0	2/ 0
lex-00 last	173/ 93	135/ 122	24/ 7	31/ 18	15/ 10	3/ 1
lex-02 2	139/ 88	147/ 129	19/ 7	27/ 17	13/ 9	0/ 0
lex-02 last	136/ 87	148/ 132	18/ 7	27/ 17	19/ 12	0/ 0
lex-60 1	54 / 50	68/ 66	25/ 7	11/ 9	85/ 38	51/ 7
lex-60 last	59 / 55	79/ 76	20/ 8	9/ 8	87/ 44	54/ 7

	all guesses / correct guesses			
iter.	copula	VPP.n	others	total
possible	**90**	**64**	**40**	**584**
random	69 / 60	67/ 51	34/ 10	560/ 336
unlex 2	96 / 77	68/ 52	32/ 12	561/ 384
unlex 60	102/ 78	47/ 8	116/ 23	580/ 294
unlex 100	102/ 78	26/ 21	124/ 25	560/ 305
lex-00 1	75 / 65	72/ 54	25/ 9	561/ 370
lex-00 last	79 / 67	72/ 54	31/ 10	563/ 382
lex-02 2	107/ 78	60/ 48	52/ 18	564/ 394
lex-02 last	106/ 78	60/ 47	50/ 17	564/ 397
lex-60 1	115/ 78	17/ 15	140/ 25	566/ 295
lex-60 last	114/ 78	20/ 18	124/ 24	566/ 318

Table 4.4: Absolute numbers of guesses and corrects for individual frames

In Table 4.4, we compare the absolute number of parser guesses and corrects of individual frames for our previously described training regime and the different resulting models. The table contains guesses and corrects of all frames that occur more than ten times in the annotator's markup. Combined measures for the set of frames that occur less than ten times in the markup are given in the column headed by *others*. The annotator's markup is indicated in the row labelled *possible*. The row labelled *random* indicates the figures for the unlexicalized model resulting from randomly initialized rule frequencies. We chose several key models of unlexicalized and lexicalized training that are indicated in the first column by iteration numbers. For unlexicalized training, we chose the the best model in iteration 2, the worst model in iteration 60, and the last model, which is already beyond the mathematical overtraining threshold. For lexicalized training from a grammar with random frequency rules, we show the figures for the worst model immediately after lexicalization (*lex-00*), and the last model. In the best lexicalized training, we focus on the second and the last model (here, the start before lexicalization is *unlex 2*). The figures for the lexicalized training stuck in a bad state (*lex-60*) are again from immediately after lexicalization and from the last iteration.

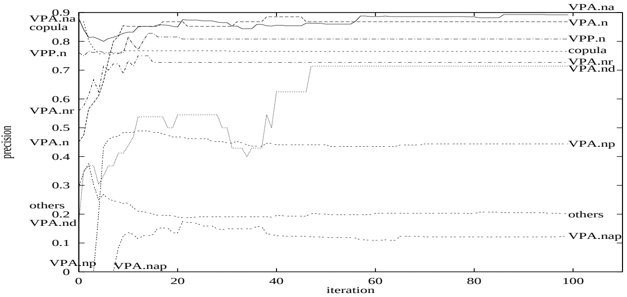


Figure 4.10: Precision of individual frames in unlexicalized training

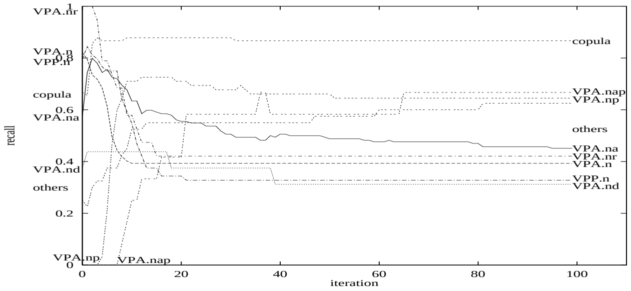


Figure 4.11: Recall of individual frames in unlexicalized training

In addition to the absolute figures of correct and incorrect guesses, we plot overall precision and recall results for individual frames in the course of unlexicalized training (Figures 4.10 and 4.11) and for those in lexicalized training starting from the best unlexicalized model (Figures 4.12 and 4.13).

unlex. The severe loss in precision following the second iteration during lexicalized training described in section 4.6.3 is due to a loss in decision mass for intransitive and transitive frames. Of 131 correctly recognized transitive frames in the second iteration more than 40 are lost in further training. Of 86 correct guesses of intransitive frames 40 are lost.

The loss with respect to intransitive frames is surprising given that the initial grammar is biased towards intransitive frames as witnessed by the figures for the random model (more than 38% of the total guesses are guesses of intransitive frames). Although the revision of intransitive guesses is desirable, it oversteps the mark of 117 in the annotator's handicap by far (213 guesses in random, 152 guesses in *unlex 2*, and 53 guesses in *unlex 60* and *100*). In the first iterations, intransitive guesses are changed to transitive and copula guesses. Concerning revisions to copula, we found a systematic error. Past participle forms of verbs are always associated with a tag list that in addition to the past participle tag VVPP also contains the tags ADJ.Pred for predicative and ADJ.Adv for adverbial adjectives. Given the ambiguous tag list, verbs forming their perfect tense forms with the auxiliary *'sein'* (*'to be'*) allow for an analysis of *'sein'* as a perfect tense auxiliary and also as a copula verb requiring a predicative. Similarly, the revision of passive frames to copula constructions in later iterations is to be explained by reanalysis of the passive auxiliary *'werden'* as a copula verb (cf. the decrease in guesses of VPP.n in *unlex 60* and *100*).

Apart from the shift to copula constructions, the dislike for intransitive and transitive is balanced by the tendency to choose frames that contain a PP-argument. The almost entire initial lack of PP-frame guesses in unlexicalized models is revised to more than 90 choices of VPA.np, more than 60 choices of VPA.nap and several low frequency PP-frames in *others*. The considerable increase in PP-frame choices contrasts with rather poor precision and recall results, less than 45% for VPA.np, less than 15% for VPA.nap, and about 20% for the PP-frames contained in *other*.

Let us add a remark on the interpretation of the precision and recall plots in figures 4.10 and 4.11. Although the majority of plots for individual frames shows a gain in precision during unlexicalized training, we noted the dramatic loss in overall precision illustrated back in Figure 4.8. As already mentioned in the above discussion of absolute numbers, this is due shift of the parser's decision mass as illustrated by the decrease in recall of intransitive, transitive, and passive frames in Figure 4.11.

lex-00/02/60. Apart from very few (mostly insignificant) exceptions, our observation concerning iterative inside-outside estimation succeeding lexicalization is confirmed by the evaluation of individual frames. The overall frame evaluation showed small but reliable precision and recall gains for frame recognition. Irrespective of different starting points, iterative lexicalized estimation similarly leads to either slightly increased or steady results for each of the individual frames with respect to absolute figures of correct guesses. The only notable negative exception is the loss of six correct choices of intransitive frames in *lex-00* accompanied by a correction from 97 to 80 wrong guesses of intransitives. The most significant gain in early lexicalization (*lex-00* and *lex-02*) is achieved for VPA.np frames. In late lexicalization, i.e. *lex-60*, lexicalized training mainly rectifies the loss of correct guesses of intransitive frames and of passive frames with a single nominative argument.

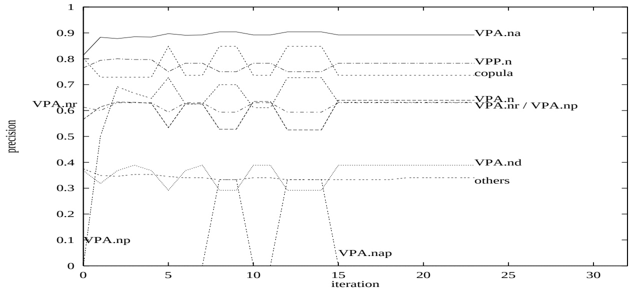


Figure 4.12: Precision of individual frames in training the best lexicalized model

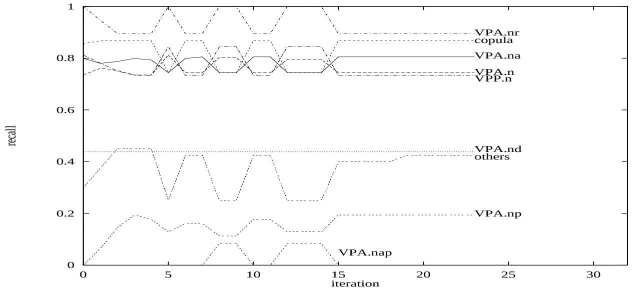


Figure 4.13: Recall of individual frames in training the best lexicalized model

The plots for precision in Figure 4.12 and recall in Figure 4.13 of individual frames in lexicalized training *lex-02* show main gains within the first four iterations. Later shifts in precision and recall up to iteration 15 are equalled out. In the above discussion of unlexicalized training, we mentioned VPP.n and the copula construction as an instance of a mutually dependent pair of frames. In the plots, this can be reidentified by precision/recall gains in the former and respective losses in the latter.

Appendix 2: Random Models

In order to give an impression of possible gains through grammar training, we did an evaluation of 50 different grammars with randomly initialized rule frequencies.

The following plots present precision and recall measures on noun chunk evaluation (Figure 4.14), frame evaluation (Figure 4.15). The precision of noun chunk recognition is in the range of 58% and 77%, frame recognition varies between 48% and 62%. We abstained from including a plot for the variability of non-PP frame recognition in different random models. It ranges from 52% to 66%.

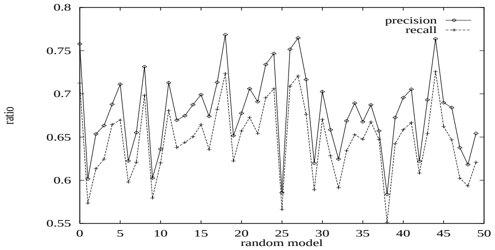


Figure 4.14: Precision/recall measures on noun chunks of 50 different unlexicalized random models

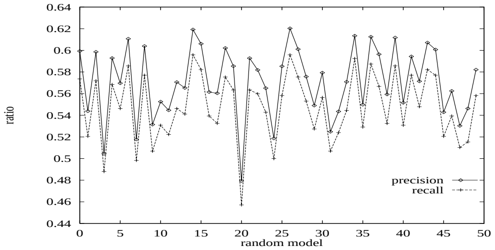


Figure 4.15: Precision/recall measures on VP frames of 50 different unlexicalized random models

Appendix 3: Different Starting Points

In order to validate our results, we chose two grammars from the 50 different rule frequency randomizations as starting points for unlexicalized and lexicalized training. Random model 20 delivers the worst precision and recall results for frames among all random states. Random model 25 yields almost identical precision and recall ratios for both noun chunks and subcategorization frames, i.e., in the context of all random models, intermediate results for noun chunks and fair results for frames. The initial grammar used in our main experiment (random grammar 0) is well above average in precision and recall for both evaluations.

Noun chunks.

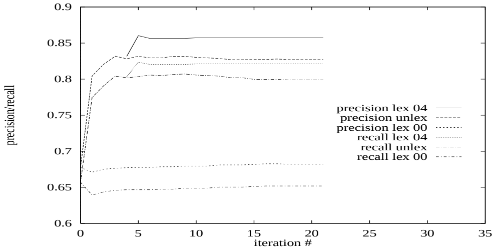


Figure 4.16: Precision/recall measures for `NC` with case (random 20)

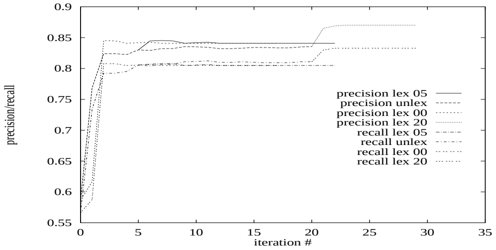


Figure 4.17: Precision/recall measures for `NC` with case (random 25)

Starting from different random grammars, precision and recall measures in noun chunk evaluation for unlexicalized training lead to similar results. Irrespective of initialization, unlexicalized training yields between 83% and 84% in precision and between 80% and 81% in recall of noun chunk recognition. A minor difference is found in the number of iterations within which the best results will be reached. Starting from a random grammar with poor precision and recall requires few more iterations of unlexicalized training in order to reach maximally possible results (cf. Figure 4.16).

For lexicalized training with different bases for lexicalization, the comparison of different starting points shows a much more diverse picture (in addition to Figure 4.18 and 4.19 see also Figure 4.8 in section 4.6.3). In contrast to the models *random 0* and *random 20*, immediate lexicalization of the random model does not produce poor results for noun chunk recognition. Furthermore, lexicalized training based on an early model of unlexicalized training may have hardly any positive effect at all as witnessed by the plots for *lex 05* in Figure 4.19. Rather, the precision results for *lex 20* in Figure 4.19 suggest the need for a more careful selection of the base for lexicalization and further lexicalized inside-outside estimation. However, since our prime interest is in good results for frame recognition, i.e. good overall results for the entire model, we have to compromise here.

Frames.

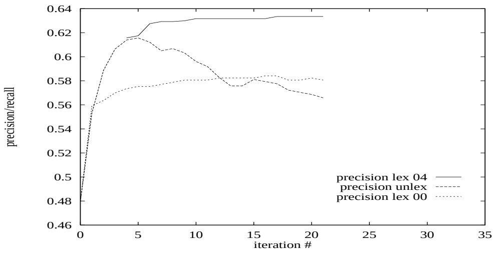


Figure 4.18: Precision measures for all verb frames (random 20)

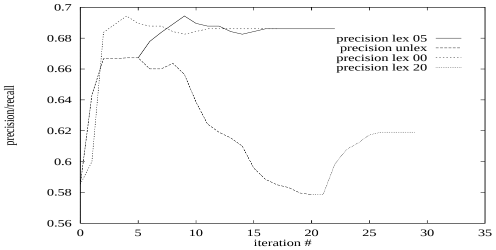


Figure 4.19: Precision measures for all verb frames (random 25)

Again, the results concerning frame recognition in iterative unlexicalized reestimation from different starting points corroborate our observations presented in section 4.6. Precision gains in initial iterations are followed by losses more or less severe depending on both precision in the initial random state and precision in the best unlexicalized model. In contrast to the observations for precision results in NC evaluation, the precision of frame recognition found for random models correlates with precision maxima. Low precision of 48% in *random 20* restricts the maximum to less than 62% within five iterations of unlexicalized training, high precision in *random 0* allows for a maximum of more than 68% within only two iterations. Irrespective of the particular starting point, 20 iterations of unlexicalized training show a uniform decrease to about 57% precision after 20 iterations.

The plots for lexicalized training with lexicalization of different base models also support our findings in section 4.6. In general, it is a good strategy to start with lexicalized training from a well-trained unlexicalized model in order to obtain best precision results. A notable exception is *lex 00* in *random 25*, where lexicalization of the random model yields similar precision for frame recognition as lexicalized training with the unlexicalized model from iteration 5 as its base.

EM-Based Clustering for NLP Applications

Mats Rooth, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil

Abstract

We present a probabilistic clustering method defining the clustering task as induction of hidden parameters of a probability model on dyadic linguistic data. Induction of unobserved class parameters is accomplished as statistical inference in the framework of maximum-likelihood estimation from incomplete data via the EM algorithm. We present illustrative examples of induced clusters from large sets of English and German data. The clustering models are evalutated on a pseudo-disambiguation task testing the power of the models to generalize over unseen data. We present two applications of the clustering models to natural language processing applications. The first task concerns the induction of semantic labels for subcategorization slots of English and German lexical verbs. Based on induced clusters of subcategorization frames and their nominal fillers, induction of labels of the respective slots is accomplished by a further application of EM. We report experiments with large sets of German and English intransitive and transitive verbs and outline a linguistic interpretation of the learned representations. A second application concerns the use of clustering models on English verb-noun combinations to select among candidate English translations of German nouns. We present an evaluation of clustering models on a pseudo-disambiguation task for noun-ambiguity. Furthermore, we evaluate the models on a gold standard extracted from a bilingual corpus. The performace of cluster-based target-language disambiguation is reported for models of various sizes and compared to simpler statistical disambiguation methods.

5.1 Introduction

Probabilistic clustering methods for natural language applications mainly focus on the following two tasks: (i) induction of smooth probability models on language data, (ii) automatic discovery of class-structure in natural language. The first task uses clustering as a solution to the problem of sparse data. In such applications the class-structure itself is not of interest, rather data clusters are consulted as general back-up sources of information when information about specific events is sparse or missing in the input. In contrast to this, for the second task we are interested in the structure of the induced clusters as a statistical semantics underlying the data in question. In the following we will present two applications of a probabilistic clustering model each of which stresses one of these tasks.

Clustering is conceptualized in our approach as induction of a class-based probability model on dyadic linguistic data—a sample of verb-noun pairs extracted from the maximal probability parses of large unannotated English and German data. Classes are introduced as hidden parameters of the probability model on verb-noun pairs. Induction of these unobserved parameters is accomplished in the framework of maximum likelihood estimation from incomplete data via the EM algorithm.

We present some illustrative examples of induced clusters and report the results of a harder evaluation of the models on a pseudo-disambiguation task testing the power of the models to generalize over unseen data.

The first application we will present is the induction of semantically annotated lexica based on induced clusters. In this application the stress is clearly on automatic discovery of class-structure. Based on clusters of subcategorization frames of verbs and their nominal fillers, induction of labels for the respective slots of subcat frames is accomplished by a further application of EM. We report results on experiments with observations derived from large English and German corpora. Furthermore, we outline an interpretation of the learned representations as theoretical-linguistic decompositional lexical entries.

The second application concentrates on the smoothing aspect of the clustering models. Here class-based probability models on English verb-noun combinations are used to disambiguate nouns senses in the context of ambiguous translations of nouns from German to English. We present an evaluation of models on a pseudo-disambiguation task for noun-ambiguity. In addition to this cheap evaluation, we present an evaluation on a gold standard which is extracted from a bilingual corpus. We report the performance of cluster-based target-language disambiguation for models of various sizes and compared to simpler methods such as plain maximum likelihood estimation on the training corpus.

5.2 EM-Based Clustering

In our clustering approach, classes are derived directly from distributional data—a sample of pairs of verbs and nouns, gathered by parsing an unannotated corpus and extracting the fillers of grammatical relations. Semantic classes corresponding to such pairs are viewed as hidden variables or unobserved data in the context of maximum likelihood estimation from incomplete data via the EM algorithm. This approach allows us to work in a mathematically well-defined framework of statistical inference, i.e., standard monotonicity and convergence results for the EM algorithm extend to our method.

The basic ideas of our EM-based clustering approach were presented in Rooth (Ms) (see also Rooth (1998)). An important property of our clustering approach is the fact that it is a “soft” clustering method, defining class membership as a conditional probability distribution over verbs and nouns. In contrast, in hard (Boolean) clustering methods such as that of Brown et al. (1992), every word belongs to exactly one class, which because of homophony is unrealistic. The foundation of our clustering model upon a probability model furthermore contrasts with the merely heuristic and empirical justification of similarity-based approaches to clustering (Dagan et al. 1998). The probability model we use can be found earlier in Pereira et al. (1993). However, in contrast to this approach, our statistical inference method for clustering is formalized clearly as an EM-algorithm. Approaches to probabilistic clustering similar to ours were presented recently in Saul and Pereira (1997) and Hofmann and Puzicha (1998). There also EM-algorithms for similar probability models have been derived, but applied only to simpler tasks not involving a combination of EM-based clustering models as in our lexicon induction experiment.

We seek to derive a joint distribution of verb-noun pairs from a large sample of pairs of verbs $v \in V$ and nouns $n \in N$. The key idea is to view v and n as conditioned on a hidden class $c \in C$, where the classes are given no prior interpretation. The semantically smoothed probability of a pair (v, n) is defined to be:

$$p(v,n) = \sum_{c \in C} p(c,v,n) = \sum_{c \in C} p(c)p(v|c)p(n|c)$$

The joint distribution $p(c, v, n)$ is defined by $p(c, v, n) = p(c)p(v|c)p(n|c)$. Note that by construction, conditioning of v and n on each other is solely made through the classes c.

In the framework of the EM algorithm (Dempster et al. 1977; McLachlan and Krishnan 1997), we can formalize clustering as an estimation problem for a latent class (LC) model as follows. We are given:

- a sample space $\mathcal{Y}$ of observed, incomplete data, corresponding to pairs from $V \times N$,
- a sample space $\mathcal{X}$ of unobserved, complete data, corresponding to triples from $C \times V \times N$,

Class 5 PROB 0.0412		0.0148	0.0084	0.0082	0.0078	0.0074	0.0071	0.0054	0.0049	0.0048	0.0047	0.0046	0.0041	0.0040	0.0040	0.0039	0.0039	0.0039	0.0039	0.0038	0.0038	0.0038	0.0037	0.0035	0.0035	0.0035	0.0034	0.0033	0.0033	0.0033	0.0033
		man	ruth	corbett	doctor	woman	athelstan	cranston	benjamin	stephen	adam	girl	laura	maggie	voice	john	harry	emily	one	people	boy	rachel	ashley	jane	caroline	jack	burun	juliet	blanche	helen	edward
0.0542	ask.as:s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	●
0.0340	nod.as:s		●	●	●	●	●	●	●	●		●	●	●		●	●	●			●	●	●	●	●	●	●	●	●	●	●
0.0299	think.as:s	●	●	●	●	●	●			●	●	●	●	●		●	●	●	●	●	●	●	●	●	●	●		●	●	●	●
0.0287	shake.aso:s	●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●		●	●	●	●	●	●	●	●	●		●	●
0.0264	smile.as:s	●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	●	●	●	●	●		●	●	●	●	●
0.0213	laugh.as:s	●	●		●		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
0.0207	reply.as:s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●			●				●	●
0.0167	shrug.as:s	●		●	●	●	●	●	●	●	●	●	●	●			●	●			●	●		●	●	●	●	●	●	●	●
0.0148	wonder.as:s	●	●	●	●	●	●			●	●	●				●	●	●		●	●	●	●	●	●	●	●	●	●	●	●
0.0141	feel.aso:s	●		●	●	●				●	●	●	●	●		●	●	●	●	●	●	●	●	●	●	●		●	●	●	●
0.0133	take.aso:s	●	●	●	●	●		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●
0.0121	sigh.as:s	●	●	●	●	●	●		●	●	●		●			●	●	●		●		●	●	●	●	●	●		●	●	●
0.0110	watch.aso:s	●	●	●		●	●	●	●	●	●	●		●		●	●	●	●	●	●	●		●		●	●	●		●	
0.0106	ask.aso:s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●			●		●			●	●	●
0.0104	tell.aso:s	●	●	●	●	●		●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	●	●	●	●	●	●	●	●
0.0094	look.as:s	●	●	●	●	●	●	●	●	●	●	●				●		●	●	●	●	●				●	●	●			●
0.0092	give.aso:s	●	●		●	●			●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
0.0089	hear.aso:s	●	●	●	●	●	●	●				●		●	●	●	●		●	●	●			●	●	●	●	●	●		
0.0083	grin.as:s	●	●	●	●	●	●	●	●		●	●	●	●		●	●	●			●	●	●	●	●	●	●			●	●
0.0083	answer.as:s	●	●	●	●	●	●	●	●					●	●	●	●	●	●	●	●		●		●	●		●		●	●
0.0081	explain.as:s		●		●	●			●	●	●		●	●	●	●	●	●	●		●	●	●	●				●	●	●	●
0.0079	frown.as:s	●	●		●	●				●	●	●	●				●	●	●	●	●	●	●	●	●					●	●
0.0076	hesitate.as:s	●	●		●	●				●	●		●		●	●	●	●	●		●	●	●		●	●		●			●
0.0074	stand.as:s	●	●	●	●	●	●	●		●		●	●		●	●	●		●	●	●	●	●		●	●		●			
0.0066	continue.as:s	●	●	●	●		●	●	●	●	●	●	●		●	●	●	●	●	●	●			●				●	●	●	
0.0065	find.aso:s	●	●		●	●			●		●	●	●	●		●	●	●	●	●	●	●		●		●		●	●		●
0.0064	feel.as:s	●	●	●	●	●		●	●		●	●		●			●	●	●	●	●	●	●	●	●	●		●	●		●
0.0062	sit.as:s	●	●	●	●	●	●	●	●			●	●	●		●	●	●	●	●	●	●		●				●	●		●
0.0062	agree.as:s	●	●	●	●		●		●	●	●	●	●	●		●		●	●	●	●	●	●	●		●	●	●	●		●
0.0056	cry.as:s	●	●	●		●		●		●	●	●	●		●				●	●	●	●		●		●					

Figure 5.1: English class 5: communicative action

- a set $X(y) = \{x \in \mathcal{X} \mid x = (c, y),\ c \in C\}$ of complete data related to the observation y,
- a complete-data specification $p_\theta(x)$, corresponding to the joint probability $p(c, v, n)$ over $C \times V \times N$, with parameter-vector $\theta = \langle \theta_c, \theta_v{}^c, \theta_n{}^c | c \in C, v \in V, n \in N \rangle$,
- an incomplete data specification $p_\theta(y)$ which is related to the complete-data specification as the marginal probability $p_\theta(y) = \sum_{X(y)} p_\theta(x)$.

The EM algorithm is directed at finding a value $\hat{\theta}$ of θ that maximizes the incomplete-data log-likelihood function L as a function of θ for a given sample $\mathcal{Y}$, i.e.,

$$\hat{\theta} = \arg\max_{\theta} L(\theta) \text{ where } L(\theta) = \ln \prod_{y} p_\theta(y).$$

As prescribed by the EM algorithm, the parameters of $L(\theta)$ are estimated indirectly by proceeding iteratively in terms of complete-data estimation for the auxiliary function $Q(\theta; \theta^{(t)})$, which is the conditional expectation of the complete-data log-likelihood $\ln p_\theta(x)$ given the observed data y and the current fit of the parameter values $\theta^{(t)}$ (E-step). This auxiliary function is iteratively maximized as a function of θ (M-step), where each iteration is defined by the map

$$\theta^{(t+1)} = M(\theta^{(t)}) = \arg\max_{\theta} Q(\theta;\theta^{(t)})$$

Note that our application is an instance of the EM-algorithm for context-free models (Baum et al. 1970; Baker 1979), from which the following particularly simple reestimation formulae can be derived. Let $x = (c, y)$ for fixed c and y, and $f(y)$ be the frequency of y in the training sample. Then

$$\begin{array}{rcl} M(\theta_{vc}) & = & \dfrac{\sum_{y\in\{v\}\times N} f(y)p_\theta(x|y)}{\sum_y f(y)p_\theta(x|y)}, \\ M(\theta_{nc}) & = & \dfrac{\sum_{y\in V\times\{n\}} f(y)p_\theta(x|y)}{\sum_y f(y)p_\theta(x|y)}, \\ M(\theta_{c}) & = & \dfrac{\sum_y f(y)p_\theta(x|y)}{|\mathcal{Y}|}. \end{array}$$

Intuitively, the conditional expectation of the number of times a particular v, n, or c choice is made during the derivation is prorated by the conditionally expected total number of times a choice of the same kind is made. As shown by Baum et al. (1970), every such maximization step increases the log-likelihood function L, and a sequence of re-estimates eventually converges to a (local) maximum of L.

5.3 Clustering Examples

In the following, we will present some examples of induced clusters. In one experiment the input to the clustering algorithm was a training corpus of 1,178,698 tokens (608,850 types) of English verb-noun pairs participating in the grammatical relations of intransitive and transitive verbs and their subject and object fillers. The data were gathered from the maximal-probability parses the head-lexicalized probabilistic context-free grammar of Carroll and Rooth (1998) gave for the British National Corpus (117 million words).

Fig. 5.1 shows an induced semantic class out of a model with 35 classes. At the top are listed the 30 most probable nouns in the $p(n|5)$ distribution and their probabilities, and at left are the 30 most probable verbs in the $p(v|5)$ distribution where 5 is the class index. Those verb-noun pairs which were seen in the training data appear with a dot in the class matrix. Verbs with suffix $.as:s$ indicate the subject slot of an active intransitive. Similarly $.aso:s$ denotes the subject slot of an active transitive, and $.aso:o$ denotes the object slot of an active transitive. Thus v in the above discussion actually consists of a combination of a verb with a subcat frame slot $as:s$, $aso:s$, or $aso:o$. Induced classes often have a basis in lexical semantics; class 5 can be interpreted as clustering agents, denoted by proper names, "man", and "woman", together with verbs denoting *communicative action*. Fig. 5.2 shows a cluster involving verbs of *scalar change* and things which can move along scales. Fig. 5.9 can be interpreted as involving different *dispositions* and modes of their execution.

Class 17 PROB 0.0265		0.0379	0.0315	0.0313	0.0249	0.0164	0.0143	0.0110	0.0109	0.0105	0.0103	0.0099	0.0091	0.0089	0.0088	0.0082	0.0077	0.0073	0.0071	0.0071	0.0070	0.0068	0.0067	0.0065	0.0065	0.0058	0.0057	0.0057	0.0054	0.0051	0.0050
		number	rate	price	cost	level	amount	sale	value	interest	demand	chance	standard	share	risk	profit	pressure	income	performance	benefit	size	population	proportion	temperature	tax	fee	time	power	quality	supplely	money
0.0437	increase.as:s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
0.0392	increase.aso:o		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●		●	●	●	●	●
0.0344	fall.as:s	●	●	●	●	●	●	●	●		●	●	●	●		●	●	●	●			●	●	●	●		●		●	●	
0.0337	pay.aso:o	●	●	●		●	●		●	●	●			●				●	●	●			●			●	●				●
0.0329	reduce.aso:o	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
0.0257	rise.as:s	●	●	●	●	●	●	●	●		●		●	●	●	●	●	●		●	●	●	●	●	●		●	●	●	●	
0.0196	exceed.aso:o	●	●	●	●	●	●		●	●	●	●	●			●		●		●	●	●	●		●	●	●	●	●	●	
0.0177	exceed.aso:s	●	●	●	●	●	●	●	●	●	●			●	●	●	●	●	●	●	●	●	●	●	●		●			●	●
0.0169	affect.aso:o	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●		●	●	●	●	●	
0.0156	grow.as:s	●						●	●	●	●			●	●	●	●	●		●	●	●	●					●			●
0.0134	include.aso:s	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	●	●	●		●	●			●	●	●	●	●	●
0.0129	reach.aso:s	●	●	●	●	●	●	●		●	●			●		●	●		●	●		●	●	●			●	●	●	●	●
0.0120	decline.as:s	●	●	●	●	●		●	●	●	●		●	●	●	●					●	●	●	●					●	●	
0.0102	lose.aso:o	●	●		●		●	●	●	●		●	●	●	●	●	●	●		●		●	●	●		●	●	●	●		●
0.0099	act.aso:s		●	●				●					●							●					●			●		●	
0.0099	improve.aso:o	●	●			●			●			●	●	●		●			●								●	●	●		
0.0088	include.aso:o	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
0.0088	cut.aso:o	●	●	●	●	●		●	●			●	●		●	●		●		●					●	●	●	●			
0.0080	show.aso:o	●	●		●	●	●	●	●	●	●		●	●	●	●	●	●	●	●	●	●	●				●	●	●		
0.0078	vary.as:s	●	●	●	●	●	●		●	●	●		●						●		●		●	●			●		●		
0.0072	give.aso:o	●	●	●	●	●	●		●	●		●	●	●	●	●	●	●	●	●	●	●	●	●			●	●	●		●
0.0071	carry.aso:o	●	●	●	●	●	●			●			●		●			●		●		●	●		●	●		●	●		●
0.0068	improve.as:s	●	●	●		●		●	●			●	●			●			●								●	●	●		
0.0066	have.as:s	●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	●	●	●		●	●	●	●	●	●	●
0.0066	produce.aso:o	●	●		●	●	●	●	●	●	●		●	●		●	●	●	●	●	●		●				●	●	●	●	●
0.0066	get.aso:o	●	●	●	●	●	●	●	●	●	●		●	●		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
0.0064	raise.aso:o	●		●	●	●	●	●		●	●	●	●	●	●	●	●	●	●	●				●	●				●	●	●
0.0063	mean.aso:o	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●	●		●		●	●	●	●	●
0.0062	receive.aso:o	●		●	●	●	●		●	●	●	●	●	●		●		●	●	●		●			●	●	●	●	●	●	●
0.0058	stand.aso:o		●									●			●												●				●

Figure 5.2: English class 17: scalar change

In another experiment, we extracted 418,290 tokens (318,086 types) of pairs of German verbs or adjectives and grammatically related nouns from maximal-probability parses; the corpus parsed was a 4.1 million word corpus of 450,000 German subordinate clauses extracted from a 200 million word corpus of German newspapers. The lexicalized statistical model for German is described in Beil et al. (1998).

The figures 5.8 and 5.3 show two classes out of a model with 35 classes. On the left and at the top are listed the 30 highest probable verb/adjective predicates and nouns appearing as fillers of the verb/adjective slots, ordered according to their probability given the class. Verbal predicates are annotated with subcategorization slots, e.g., *liegen-VPA.np:NP.Nom* denotes the nominative noun-phrase filler of the subject-slot of an active verb *liegen* subcategorizing for a nominative and a prepositional phrase. *tragen-VPA.na:NP.Akk* is the accusative noun-

Class 26 PROB 0.0223		0.0379	0.0226	0.0182	0.0171	0.0137	0.0133	0.0129	0.0086	0.0079	0.0079	0.0073	0.0072	0.0071	0.0067	0.0064	0.0063	0.0061	0.0057	0.0056	0.0053	0.0051	0.0047	0.0046	0.0039	0.0038	0.0037	0.0037	0.0036	0.0036	0.0035
		Ergebnis	Preis	Menge	Anteil	St"uck	Zahl	Gewinn	Kritiker	B"urgerMeister	Angst	Umsatz	Einnahme	Zins	Schicksal	Bus	NachFrage	Ertrag	Figur	Verlust	Frucht	ArbeitsLosigkeit	Kost	Last	WahlErgebnis	Temperatur	Solidarit"at	InflationsRate	Abschlu"s	Import	unterSchrift
0.0601	liegen-VPA.np:NP.Nom	●	●	●	●	●	●	●		●				●	●			●		●			●			●		●			
0.0351	tragen-VPA.na:NP.Akk			●	●									●						●	●		●	●							
0.0230	steigen-VPA.n:NP.Nom		●	●	●		●					●	●	●			●			●		●				●		●			
0.0213	sagen-VPA.n:NP.Nom								●	●																					
0.0182	gering-ADJ		●	●	●		●						●	●				●		●		●	●			●		●			
0.0135	sinken-VPA.n:NP.Nom		●		●			●					●	●			●	●								●					
0.0135	steigen-VPA.np:NP.Nom				●		●										●	●		●		●				●					
0.0119	erkl"aren-VPA.n:NP.Nom									●																					
0.0100	positiv-ADJ	●																●													
0.0091	zur"uck+gehen-VPA.np:NP.Nom				●		●	●				●	●					●										●		●	
0.0087	sinken-VPA.np:NP.Nom				●		●					●		●												●	●			●	
0.0082	sp"at-ADJ												●					●													
0.0077	wirken-VPA.n:NP.Nom																														
0.0071	"andern-VPA.na:NP.Akk														●																
0.0071	regional-ADJ																														
0.0067	Erfolgreich-ADJ																												●		
0.0065	best"atigen-VPA.n:NP.Nom									●	●																				
0.0057	steigen-ADJ		●				●	●				●		●			●					●	●					●			
0.0056	an+steigen-VPA.np:NP.Nom		●		●		●					●										●				●		●			
0.0054	formulieren-VPA.na:NP.Nom									●																					
0.0052	b"ose-ADJ														●																
0.0051	an+steigen-VPA.n:NP.Nom				●			●									●					●	●			●					
0.0051	sitzen-VPA.n:NP.Nom										●																				
0.0049	"ubersteigen-VPA.na:NP.Nom	●	●		●		●					●	●	●			●			●			●					●			
0.0045	ein+setzen-VPP.n:NP.Nom															●															
0.0045	an+erkennen-VPA.na:NP.Akk	●																					●		●				●		
0.0045	entdecken-VPA.nap:NP.Akk										●								●												
0.0043	zu+nehmen-VPA.np:NP.Nom	●			●						●	●	●				●					●								●	
0.0043	betrachten-VPA.na:NP.Akk						●																								
0.0043	senken-VPP.n:NP.Nom		●	●			●							●			●											●			

Figure 5.3: German class 26: scalar change

phrase filler of the object slot of the transitive verb ***tragen***, ***steigen-VPA.n:NP.Nom*** denotes the nominative filler of the subject slot of the intransitive verb ***steigen***. Clearly, due to the smaller size of the German input data compared to the English data, German classes are less dense than the English counterparts.

Fig. 5.8 shows a class which can be interpreted as *govermental/public authority*, involving nouns such as *police force* and *public prosecutor's office*. Fig. 5.3 shows a cluster involving scalar motion verbs and things which can move along scales.

5.4 Evaluation of Clustering Models

5.4.1 Pseudo-Disambiguation

We evaluated our clustering models on a pseudo-disambiguation task similar to that performed in Pereira et al. (1993), but differing in detail. The task is to judge which of two verbs v and v' is more likely to take a given noun n as its argument where the pair (v, n) has been cut out of the original corpus and the pair (v', n) is constructed by pairing n with a randomly chosen verb v' such that the combination (v', n) is completely unseen. Thus this test evaluates how well the models generalize over unseen verbs.

The data for this test were built as follows. We constructed an evaluation corpus of (v, n, v') triples from a test corpus of 3,000 types of (v, n) pairs which were randomly cut out of the original corpus of 1,280,712 tokens, leaving a training corpus of 1,178,698 tokens. Each noun n in the test corpus was combined with a verb v' which was randomly chosen according to its frequency such that the pair (v', n) did appear neither in the training nor in the test corpus. However, the elements v, v', and n were required to be part of the training corpus. Furthermore, we restricted the verbs and nouns in the evaluation corpus to the ones which occurred at least 30 times and at most 3,000 times with some verb-functor v in the training corpus. The resulting 1,337 evaluation triples were used to evaluate a sequence of clustering models trained from the training corpus.

The clustering models we evaluated were parameterized in starting values of the training algorithm, in the number of classes of the model, and in the number of iteration steps, resulting in a sequence of $3 \times 10 \times 6$ models. Starting from a lower bound of 50 % for randomly initialized models, accuracy was calculated as the number of times the model decided for $p(n|v) \geq p(n|v')$ out of all choices made. Fig. 5.4 shows the evaluation results for models trained with 50 iterations, averaged over starting values, and plotted against class cardinality. Different starting values had an effect of ± 2 % on the performance of the test. We obtained a value of about 80 % accuracy for models between 25 and 100 classes. Models with more than 100 classes show a small but stable overfitting effect.

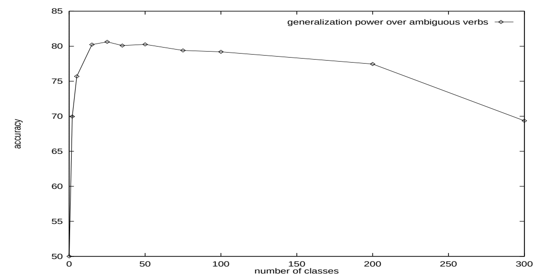


Figure 5.4: Evaluation of English models on pseudo-disambiguation task

The German models were evaluated in a similar way. An evaluation corpus of 886 (v, n, v') triples was extracted from the original corpus of 428,446 verb/adjective-noun tokens, leaving 418,290 tokens for training a sequence of clustering models. Again, the models were parameterized in starting values, number of classes and iteration steps, resulting in a sequence of $3 \times 11 \times 20$ models. Fig. 5.5 shows the evaluation results for models trained with 100 iterations, averaged over starting values, and plotted against class cardinality. We obtained an accuracy of over 75 % for models up to 35 classes. Different starting values had an effect of ± 2 % on the evaluation results. For models with more than 50 classes again a small overfitting effect can be seen.

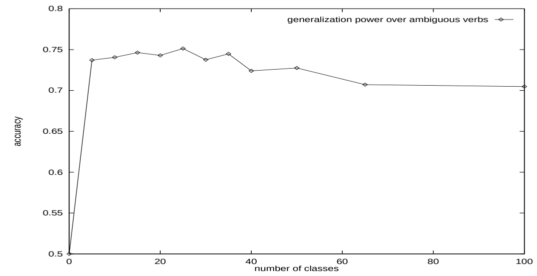


Figure 5.5: Evaluation of German models on pseudo-disambiguation task

5.4.2 Smoothing Power

A second experiment addressed the smoothing power of the model by counting the number of (v, n) pairs in the set $V \times N$ of all possible combinations of verbs and nouns which received

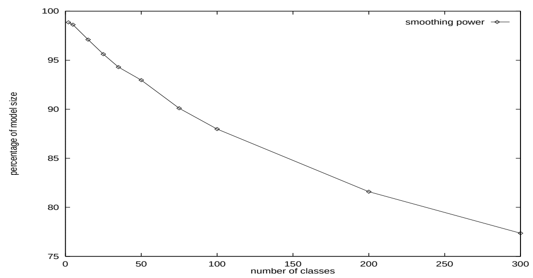


Figure 5.6: Evaluation of English models on smoothing task

a positive joint probability by the model. The $V \times N$-space for the above clustering models included about 425 million (v, n) combinations; we approximated the smoothing size of a model by randomly sampling 1,000 pairs from $V \times N$ and returning the percentage of positively assigned pairs in the random sample. Fig. 5.6 plots the smoothing results for the above models against the number of classes. Starting values had an influence of ± 1 % on performance. Given the proportion of the number of types in the training corpus to the $V \times N$-space, without clustering we have a smoothing power of 0.14 % whereas for example a model with 50 classes and 50 iterations has a smoothing power of about 93 %.

Corresponding to the maximum likelihood paradigm, the number of training iterations had a decreasing effect on the smoothing performance whereas the accuracy of the pseudo-disambiguation was increasing in the number of iterations. We found a number of 50 iterations to be a good compromise in this trade-off.

For German models we observed a baseline smoothing power of 0.012 % which is the relation of the number of types in the German training corpus to the 2.5 billion combinations in the $V \times N$-space for the German experiments. Despite of the fact that this baseline is 10 times smaller than the baseline for the English models, we have a smoothing power of about 32 % for models with 25 classes, which were best in terms of the pseudo-disambiguation task. This is shown in Fig. 5.7. The best compromise in terms of iterations was a number of 100 iterations for the German experiments.

5.5 Application to Lexicon Induction Based on Latent Classes

5.5.1 Motivation

An important challenge in computational linguistics concerns the construction of large-scale computational lexicons for the numerous natural languages where very large samples of lan-

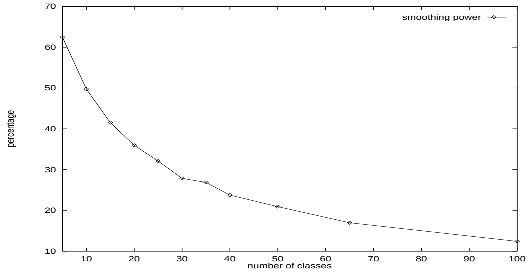


Figure 5.7: Evaluation of German models on smoothing task

guage use are now available. Resnik (1993) initiated research into the automatic acquisition of semantic selectional restrictions. Ribas (1994) presented an approach which takes into account the syntactic position of the elements whose semantic relation is to be acquired. However, those and most of the following approaches require as a prerequisite a fixed taxonomy of semantic relations. This is a problem because (i) entailment hierarchies are presently available for few languages, and (ii) we regard it as an open question whether and to what degree existing designs for lexical hierarchies are appropriate for representing lexical meaning. Both of these considerations suggest the relevance of inductive and experimental approaches to the construction of lexicons with semantic information.

This section presents a method for automatic induction of semantically annotated subcategorization frames from unannotated corpora. We use a statistical subcat-induction system which estimates probability distributions and corpus frequencies for pairs of a head and a subcat frame (Carroll and Rooth 1998). The statistical parser can also collect frequencies for the nominal fillers of slots in a subcat frame. The induction of labels for slots in a frame is based upon estimation of a probability distribution over tuples consisting of a class label, a selecting head, a grammatical relation, and a filler head. The class label is treated as hidden data in the EM-framework for statistical estimation.

5.5.2 Probabilistic Labeling with Latent Classes using EM-estimation

To induce latent classes for the subject slot of a fixed intransitive verb the following statistical inference step was performed. Given a latent class model $p_{LC}(\cdot)$ for verb-noun pairs, and a sample $n_1, \ldots, n_M$ of subjects for a fixed intransitive verb, we calculate the probability of an arbitrary subject $n \in N$ by:

$$p(n) = \sum_{c \in C} p(c,n) = \sum_{c \in C} p(c) p_{LC}(n|c).$$

Class 14 PROB 0.0283		0.0877	0.0303	0.0187	0.0179	0.0161	0.0154	0.0151	0.0150	0.0130	0.0112	0.0094	0.0088	0.0086	0.0085	0.0073	0.0066	0.0066	0.0064	0.0061	0.0061	0.0059	0.0058	0.0057	0.0053	0.0051	0.0051	0.0043	0.0042	0.0041	0.0041
		Polizei	deutschLand	Nation	SPD	USA	Koalition	Sprecher	Verein	StaatsAnwaltschaft	Beh"orde	Bonn	firm	UNO	BundesAmt	Veranstalter	Blatt	Konzern	Magistrat	Sender	oberB"urgerMeister	BundesBank	B"urgerMeister	fern+sehen	Nato	Zeitung	N"ahe	nachrichtenAgentur	PolizeiSprecher	LandesRegierung	rundFunk
0.0581	mit+teilen-VPA.n:NP.Nom	●						●	●		●				●	●		●	●	●			●					●	●		●
0.0220	berichten-VPA.n:NP.Nom	●			●			●			●				●		●			●		●		●		●		●	●		●
0.0141	vereinen-ADJ		●	●																											
0.0120	sagen-VPA.n:NP.Nom	●						●								●							●								
0.0114	Bremer-ADJ	●											●							●			●							●	
0.0113	mit+teilen-VPA.np:NP.Nom			●				●		●	●								●			●	●								
0.0095	erkl"aren-VPA.n:NP.Nom	●						●	●						●						●		●						●		
0.0070	hessisch-ADJ	●			●																									●	●
0.0065	unmittelbar-ADJ																										●				
0.0064	fordern-VPA.na:NP.Nom		●		●	●			●			●							●							●				●	
0.0063	machen-VPA.nad:NP.Nom	●			●	●			●	●									●												
0.0061	verzichten-VPA.np:NP.Nom		●		●	●	●				●			●											●					●	
0.0060	melden-VPA.n:NP.Nom							●							●									●		●		●			●
0.0059	Berliner-ADJ	●							●	●	●									●						●				●	
0.0055	spielen-VPA.nap:NP.Nom		●			●																									
0.0052	Westdeutsch-ADJ												●					●													●
0.0046	statistisch-ADJ														●																
0.0043	meinen-VPA.n:NP.Nom					●		●											●				●								
0.0043	auf+nehmen-VPA.nap:NP.Nom	●	●		●	●				●		●		●																	
0.0042	w"unschen-VPA.nar:NP.Akk																														
0.0041	vor+stellen-VPA.nar:NP.Nom				●		●		●													●			●					●	
0.0041	haben-VPA.nap:NP.Nom	●	●		●	●	●				●				●	●						●		●	●					●	
0.0041	betonen-VPA.n:NP.Nom	●														●					●		●								
0.0040	zust"andig-ADJ									●	●												●								
0.0039	"ubernehmen-VPA.na:NP.Nom	●			●	●	●		●		●	●		●				●	●												
0.0038	nieders"achsisch-ADJ	●			●																									●	
0.0036	"ortlich-ADJ	●							●		●															●					
0.0035	best"atigen-VPA.n:NP.Nom							●		●	●										●		●						●		
0.0035	erz"ahlen-VPA.n:NP.Nom							●																							
0.0033	ein+treffen-VPA.n:NP.Nom	●																								●					

Figure 5.8: German class 14: governmental/public authority

The estimation of the parameter-vector $\theta = \langle \theta_c | c \in C \rangle$ can be formalized in the EM framework by viewing $p(n)$ or $p(c,n)$ as a function of θ for fixed $p_{LC}(.)$. The re-estimation formulae resulting from the incomplete data estimation for these probability functions have the following form ($f(n)$ is the frequency of n in the sample of subjects of the fixed verb):

$$M(\theta_c) = \frac{\sum_{n \in N} f(n) p_\theta(c|n)}{\sum_{n \in N} f(n)}$$

A similar EM induction process can be applied also to pairs of nouns, thus enabling induction of latent semantic annotations for transitive verb frames. Given a LC model $p_{LC}(\cdot)$ for verb-noun pairs, and a sample $(n_1, n_2)_1, \ldots, (n_1, n_2)_M$ of noun arguments (n_1 subjects, and n_2 direct objects) for a fixed transitive verb, we calculate the probability of its noun argument pairs by:

$$\begin{aligned} p(n_1, n_2) &= \sum_{c_1, c_2 \in C} p(c_1, c_2, n_1, n_2) \\ &= \sum_{c_1, c_2 \in C} p(c_1, c_2) p_{LC}(n_1|c_1) p_{LC}(n_2|c_2) \end{aligned}$$

Again, estimation of the parameter-vector $\theta = \langle \theta_{c_1 c_2} | c_1, c_2 \in C \rangle$ can be formalized in an EM framework by viewing $p(n_1, n_2)$ or $p(c_1, c_2, n_1, n_2)$ as a function of θ for fixed $p_{LC}(.)$. The re-estimation formulae resulting from this incomplete data estimation problem have the following simple form ($f(n_1, n_2)$ is the frequency of (n_1, n_2) in the sample of noun argument pairs of the fixed verb):

$$M(\theta_{c1c2}) = \frac{\sum_{n_1, n_2 \in N} f(n_1, n_2) p_\theta(c_1, c_2 | n_1, n_2)}{\sum_{n_1, n_2 \in N} f(n_1, n_2)}$$

Note that the class distributions $p(c)$ and $p(c_1, c_2)$ for intransitive and transitive models can also be computed for verbs unseen in the LC model.

5.5.3 Lexicon Induction Experiments

In a first experiment with English data we used a model with 35 classes. From maximal probability parses for the British National Corpus derived with the statistical parser of Carroll and Rooth (1998), we extracted frequency tables for intransitive verb/subject pairs and transitive verb/subject/object triples. The 500 most frequent verbs were selected for slot labeling. Fig. 5.10 shows two verbs v for which the most probable class label is 5, a class which we earlier described as *communicative action*, together with the estimated frequencies of $f(n)p_\theta(c|n)$ for those ten nouns n for which this estimated frequency is highest.

Fig. 5.11 shows corresponding data for an intransitive scalar motion sense of *increase*.

Fig. 5.12 shows the intransitive verbs which take 17 as the most probable label. Intuitively, the verbs are semantically coherent. When compared to Levin (1993)'s 48 top-level verb classes,

Class 8 PROB 0.0369		0.0385	0.0162	0.0157	0.0101	0.0073	0.0071	0.0063	0.0060	0.0060	0.0057	0.0055	0.0052	0.0052	0.0051	0.0050	0.0050	0.0048	0.0047	0.0047	0.0046	0.0041	0.0041	0.0040	0.0040	0.0039	0.0038	0.0037	0.0037	0.0036	0.0036
		change	use	increase	development	growth	effect	result	degree	response	approach	reduction	forme	condition	understanding	improvement	treatment	skill	action	process	activity	knowledge	factor	level	type	reaction	kind	difference	movement	loss	amount
0.0539	require.aso:o	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●			●	●		●
0.0469	show.aso:o	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●		●	●	●	●
0.0439	need.aso:o	●	●	●	●		●	●	●	●	●	●	●	●	●	●	●	●	●		●	●		●	●	●	●	●	●		●
0.0383	involve.aso:o	●		●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	●		●	●	●	●	●	●	●	●	●	●
0.0270	produce.aso:o	●	●	●	●	●		●	●	●	●	●	●	●	●	●			●		●	●	●	●	●	●	●	●	●	●	●
0.0255	occur.as:s	●	●	●	●	●	●	●	●	●	●	●	●	●		●			●	●	●		●	●	●	●	●	●	●	●	
0.0192	cause.aso:s	●	●	●	●	●	●	●		●		●	●	●			●		●	●	●	●	●	●	●	●	●	●	●	●	●
0.0189	cause.aso:o	●	●	●	●	●	●	●	●	●				●		●			●		●			●		●	●	●	●	●	●
0.0179	affect.aso:s	●	●	●	●	●	●	●	●	●	●	●	●	●			●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
0.0162	require.aso:s	●	●	●	●		●	●	●	●	●	●	●	●	●		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
0.0150	mean.aso:o	●	●	●	●	●	●		●		●	●	●	●	●	●	●		●	●	●	●		●	●		●	●	●	●	●
0.0140	suggest.aso:o	●	●	●	●	●	●	●	●	●	●	●	●	●		●	●		●	●	●	●	●	●	●		●	●	●		●
0.0138	produce.aso:s	●	●	●	●	●	●	●	●	●	●	●	●	●			●		●	●	●		●	●	●	●	●	●	●	●	●
0.0109	demand.aso:o	●	●	●	●			●	●	●	●		●		●	●	●	●	●		●	●		●			●				●
0.0109	reduce.aso:s	●	●	●	●	●			●	●	●	●	●				●	●	●	●	●	●	●	●					●	●	●
0.0097	reflect.aso:o	●	●	●	●	●	●		●	●	●	●	●	●	●	●		●		●	●	●	●	●	●	●	●	●	●	●	●
0.0092	involve.aso:s	●	●			●	●	●		●	●	●	●		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●
0.0091	undergo.aso:o	●	●	●	●		●		●					●		●	●			●						●	●			●	●
0.0086	increase.aso:s	●	●	●	●	●	●	●		●	●	●				●	●	●	●	●		●	●	●	●				●	●	●
0.0081	allow.aso:o	●	●	●	●	●			●		●	●			●	●	●		●	●				●			●		●	●	●
0.0079	include.aso:o			●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●		●	●	●	●	●
0.0075	make.aso:s	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●
0.0075	encourage.aso:o	●	●	●	●	●			●	●	●	●			●	●		●		●	●			●	●		●		●	●	●
0.0073	saw.aso:o	●	●	●	●	●	●	●	●		●	●	●	●	●	●			●					●	●	●	●	●	●		●
0.0072	create.aso:s	●	●	●	●	●	●	●		●	●	●	●	●		●			●	●	●		●	●	●	●	●	●	●	●	
0.0070	affect.aso:o	●	●		●	●	●	●	●	●	●	●	●	●	●				●	●	●			●	●	●	●	●	●		●
0.0069	imply.aso:o	●	●	●		●			●		●	●	●	●	●	●			●	●		●		●	●		●	●			●
0.0068	achieve.aso:o	●	●	●		●	●	●	●	●		●	●		●	●		●		●				●		●	●		●	●	●
0.0066	find.aso:o	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●	●
0.0062	describe.aso:o	●	●	●	●	●	●	●		●	●		●	●				●	●	●	●	●	●	●	●	●	●	●	●	●	●

Figure 5.9: English class 8: dispositions

we found an agreement of our classification with her class of "verbs of changes of state" except for the last three verbs in the list in Fig. 5.12 which is sorted by probability of the class label.

Fig. 5.13 shows the most probable pair of classes for *increase* as a transitive verb, together with estimated frequencies for the head filler pair. Note that the object label 17 is the class found with intransitive scalar motion verbs; this correspondence is exploited in the next section.

Further experiments were done with two German models with 35 and 50 classes respectively. The data for these experiments were extracted from the maximal probability parses of a 4.1 million word corpus of German subordinate clauses, parsed with the lexicalized probabilistic grammar of Beil et al. (1998). Fig. 5.14 shows the subjects of the intransitive verb *bekanntgeben* (make public). The nouns are classified with probability 0.999999 to class 14, which was described in Sect. 5.3 as class of *govermental/public-authority*. The numbers in the column show the estimated frequencies of the subject fillers.

Fig. 5.15 shows the subjects of the intransitive verb *steigen* (rise) which belong with

blush 0.982975		*snarl* 5	0.962094
constance	3	mandeville	2
christina	3	jinkwa	2
willie	2.99737	man	1.99859
ronni	2	scott	1.99761
claudia	2	omalley	1.99755
gabriel	2	shamlou	1
maggie	2	angalo	1
bathsheba	2	corbett	1
sarah	2	southgate	1
girl	1.9977	ace	1

Figure 5.10: Lexicon entries: *blush*, *snarl*

increase 17	0.923698		
number	134.147	proportion	3.8699
demand	0.7322	size	22.8108
pressure	0.5844	rate	0.9593
temperature	25.9691	level	0.7651
cost	3.9431	price	17.9996

Figure 5.11: Scalar motion *increase*.

probability 0.67273 to class 26 which was interpreted in Sect. 5.3 as a class of *gradation/scalar change*.

Similar to the English experiments we observe semantic uniformity in the verbs of scalar change. Fig. 5.16 shows 10 intransitive verbs which take class 14 of a 50-classes model (corresponding to class 26 of the 35-class model) as the most probable class to label their respective subject slots. On the basis the most probable class labels these verbs can be summarized as scalar motion verbs. When compared to linguistic classifications of verbs given by Schuhmacher (1986), we found an agreement of our classification with the class of "einfache Änderungsverben" (simple verbs of change) except for the verbs *anwachsen* (increase) and *stagnieren* (stagnate) which were not classified there at all.

An example of the two most probable subject-object class pairs of a transitive verb, *senken* (lower) is shown in Fig. 5.17. Class 14 has been introduced before as *govermental/public authority* and class 26 as *gradation/scalar change*.

Fig. 5.18 shows the transitive verb *dauern* (last/go on) selecting the class-pair (0, 10) with probability 0.957095 as semantic label for its subject and object slots. Class 0 can be interpreted as *project/action-class* and class 10 as class of *time*.

0.977992 decrease	0.560727 drop
0.948099 double	0.476524 grow
0.923698 increase	0.42842 vary
0.908378 decline	0.365586 improve
0.877338 rise	0.365374 climb
0.876083 soar	0.292716 flow
0.803479 fall	0.280183 cut
0.672409 slow	0.238182 mount
0.583314 diminish	

Figure 5.12: Scalar motion verbs

increase (8, 17)	0.3097650
development - pressure	2.3055
fat - risk	2.11807
communication - awareness	2.04227
supplementation - concentration	1.98918
increase - number	1.80559

Figure 5.13: Transitive *increase* with estimated frequencies for filler pairs.

5.5.4 Linguistic Interpretation of Latent Class Labels

In some linguistic accounts, multi-place verbs are decomposed into representations involving (at least) one predicate or relation per argument. For instance, the transitive causative/inchoative verb *increase,* is composed of an actor/causative verb combining with a one-place predicate in the structure on the left in Fig. 5.19. Linguistically, such representations are motivated by argument alternations (diathesis), case linking and deep word order, language acquistion, scope ambiguity, by the desire to represent aspects of lexical meaning, and by the fact that in some languages, the postulated decomposed representations are overt, with each primitive predicate corresponding to a morpheme. For references and recent discussion of this kind of theory see Hale and Keyser (1993) and Kural (1996).

We will sketch an understanding of the lexical representations induced by latent-class labeling in terms of the linguistic theories mentioned above, aiming at an interpretation which combines computational learnability, linguistic motivation, and denotational-semantic adequacy. The basic idea is that latent classes are computational models of the atomic relation symbols occurring in lexical-semantic representations. As a first implementation, consider replacing the relation symbols in the first tree in Fig. 5.19 with relation symbols derived from the latent class labeling. In the second tree in Fig 5.19, R_{17} and R_8 are relation symbols with indices derived from the labeling procedure of Sect. 5.5. Such representations can be semantically interpreted in standard ways, for instance by interpreting relation symbols as denoting

bekanntgeben 14	0.999999	(make public)
Sprecher	4	(spokesman)
Polizei	3	(police)
BundesAmt	3	(Federal Agency)
BürgerMeister	2	(mayor)
VorstandsChef	2	(Chairman of the board)
GeschäftsLeitung	2	(manager)
Vorstand	2	(board of management)
unternehmen	1.99996	(company)
WetterAmt	1	(meteorological office)
VolksBank	1	(cooperative bank)

Figure 5.14: Intransitive lexicon entry: *bekanntgeben* (make public)

steigen 2667273		(rise)
Zahl	23.333	(number)
Preis	5.895	(price)
ArbeitsLosigkeit	10.8788	(unemployment)
Lohn	9.72965	(wage)
NachFrage	6.83619	(demand)
Zins	6.80322	(interest)
Auflage	5.22654	(print run)
Beitrag	4.22577	(contribution)
Produktion	4.21641	(output)
GrundstuecksPreis	4	(price of a piece of land)

Figure 5.15: Intransitive lexicon entry: *steigen* (rise)

0.741467 ansteigen	(go up)
0.720221 steigen	(rise)
0.693922 absinken	(sink)
0.656021 sinken	(go down)
0.438486 schrumpfen	(shrink)
0.375039 zurückgehen	(decrease)
0.316081 anwachsen	(increase)
0.215156 stagnieren	(stagnate)
0.160317 wachsen	(grow)
0.154633 hinzukommen	(be added)

Figure 5.16: German intransitive scalar change verbs

senken $(14, 26)$	0.450352	(lower)
BundesBank - LeitZins	5.81457	(Federal bank - base rate)
BundesBank - Zins	2.97838	(Federal bank - interest)
superMarkt - Preis	1	(super market - price)
SommerGeschäft - Verlust	1	(summer business - loss)
BundesBank - DiskontSatz	0.99999	(Federal bank - minimum lending rate)
senken $(14, 14)$	0.147857	
BundesBank - Lombardsatz	0.999973	(Federal bank - rate on loanes on security)
StrafAndrohung - AbtreibungsQuote	0.96842	(threat of punishment - abortion rate)
StrafAndrohung - AbtreibungsZahl	0.96842	(threat of punishment - number of abortions)
FachHandel - LagerKost	0.878333	(stores - storage charges)
Harmonisierung - sozialNiveau	0.764319	(harmonization - social level)

Figure 5.17: Transitive lexicon entries for *senken* (lower)

relations between events and individuals.

Such representations are semantically inadequate for reasons given in philosophical critiques of decomposed linguistic representations; see Fodor (1998) for recent discussion. A lexicon estimated in the above way has as many primitive relations as there are latent classes. We guess there should be a few hundred classes in an approximately complete lexicon (which would have to be estimated from a corpus of hundreds of millions of words or more). Fodor's arguments, which are based on the very limited degree of genuine interdefinability of lexical items and on Putnam's arguments for contextual determination of lexical meaning, indicate that the number of basic concepts has the order of magnitude of the lexicon itself. More concretely, a lexicon constructed along the above principles would identify verbs which are labelled with the same latent classes; for instance it might identify the representations of *grab* and *touch*.

For these reasons, a semantically adequate lexicon must include additional relational con-

dauern $(0, 10)$	0.957095	(last/go on)
Entwirrung - Zeit	2	(disentanglement - time)
BuergerFrageStunde - Stunde	2	(question time - hour)
Prozess - Jahr	2	(trail - year)
schreckensZeit - Jahr	1	(scaring time - year)
ratenZahlung - Jahr	1	(buy in installments - year)

Figure 5.18: Transitive lexicon entry for *dauern* (last/to go on)

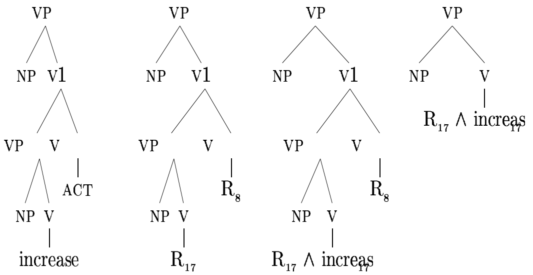


Figure 5.19: First tree: linguistic lexical entry for transitive verb *increase*. Second: corresponding lexical entry with induced classes as relational constants. Third: indexed open class root added as conjunct in transitive scalar motion *increase*. Fourth: induced entry for related intransitive *increase*.

stants. We meet this requirement in a simple way, by including as a conjunct a unique constant derived from the open-class root, as in the third tree in Fig. 5.19. We introduce indexing of the open class root (copied from the class index) in order that homophony of open class roots not result in common conjuncts in semantic representations—for instance, we don't want the two senses of *decline* exemplified in *decline the proposal* and *decline five percent* to have a common entailment represented by a common conjunct. This indexing method works as long as the labeling process produces different latent class labels for the different senses.

The last tree in Fig. 5.19 is the learned representation for the scalar motion sense of the intransitive verb *increase*. In our approach, learning the argument alternation (diathesis) relating the transitive *increase* (in its scalar motion sense) to the intransitive *increase* (in its scalar motion sense) amounts to learning representations with a common component $R_{17} \wedge \text{increase}_{17}$. In this case, this is achieved.

5.5.5 Discussion

We have proposed a procedure which maps observations of subcategorization frames with their complement fillers to structured lexical entries. We believe the method is scientifically interesting, practically useful, and flexible because:

1. The algorithms and implementation are efficient enough to map a corpus of a hundred million words to a lexicon.
2. The model and induction algorithm have foundations in the theory of parameterized families of probability distributions and statistical estimation. As exemplified in the paper, learning, disambiguation, and evaluation can be given simple, motivated formulations.
3. The derived lexical representations are linguistically interpretable. This suggests the possibility of large-scale modeling and observational experiments bearing on questions arising in linguistic theories of the lexicon.
4. Because a simple probabilistic model is used, the induced lexical entries could be incorporated in lexicalized syntax-based probabilistic language models, in particular in head-lexicalized models. This provides for potential application in many areas.
5. The method is applicable to any natural language where text samples of sufficient size, computational morphology, and a robust parser capable of extracting subcategorization frames with their fillers are available.

The pseudo-disambiguation task of Sect. 5.4 provides a hard evaluation of the learned models, allowing comparison to other methods. The discussion of linguistic interpretation suggests entirely independent linguistic evaluations of the adequacy of the learned lexicon.

For instance, by linguistic criteria it is approximately uncontroversial whether a given verb with both intransitive and transitive frames has a subject/object diathesis (as e.g. *increase* does, but *decline* doesn't). This would allow for an evaluation of a procedure (like the one proposed here) which learns a lexicon with hidden linguistic structure.

5.6 Application to Target-Language Disambiguation in Translation

5.6.1 Motivation

In most transfer-based approaches to machine translation the major part of the work is done by symbolic, i.e. non-statistical translation mechanisms. Clearly, a statistical disambiguation (SD) component supporting the symbolic transfer can be very helpful in cases of failure of the symbolic disambiguation component or cases of explosion of ambiguities. Such a supporting SD component should ideally have the following features:

- **robustness**: In order to enable a disambiguation in most cases, one of the central features of SD is its ability to assign a positive though possibly very small probability to nearly every structure under consideration.
- **efficiency**: In cases of explosion of translation ambiguities, SD must facilitate quick resolution.
- **economy**: SD must be able to resolve translation ambiguities with a minimum of information delivered from the symbolic translation component.
- **domain-specificity**: SD must be able to disambiguate translation alternatives according to their use in a specific domain.
- **portability**: A SD component for machine translation systems working with more than one source/target language pair ideally should rely only on information which is automatically obtainable from monolingual corpora.
- **accuracy**: To built a sensible probability model on ambiguous structures, the probabilities assigned to translation alternatives must be gathered by a mathematically well-defined statistical inference mechanism and proven to be useful in a task-oriented evaluation procedure.

To illustrate these features, let us have a look at some recent approaches to statistical disambiguation. One of the most influential works on target word selection in machine translation is described in Dagan and Itai (1994). The SD of Dagan and Itai (1994) uses statistics exclusively on monolingual data. The information gathered is statistics on all grammatical relations in which an ambiguous lexical item participates. In an experiment on translating Hebrew to English, their statistical model was applicable to 70 out of 100 of the ambiguous words, achieving a precision of 91 %. In a second experiment on translating German to English, their statistical model was applicable to 27 out of 54 of the ambigous words, achieving a precision of 78 %. Given the relatively low recall values of 70 % and 50 %, their SD system clearly lacks robustness. The usage of rich information on grammatical relations lets the approach seem uneconomic and presumably not very efficient and portable. However, this approach achieves high precision rates on small test corpora with low averages of 3.3 alternative translations and high averages of 1.4 correct translations.

Another important approach to SD is the work of Melamed (1997). Melamed reports 42 % precision and 35-40 % recall using a gold standard for single-best word-to-word translations (French-English). The test corpus contains several thousand items and is a part of the training corpus. Melamed's best model can be viewed as a robust, economic, domain-specific, and portable SD. Unfortunately, the size of average alternative translations is not reported. Presumably, it is high. More importantly, the models were trained by using a part of the training corpus as test set and the reported recall measures are not high. Thus, the models main drawback is its lack of accuracy.

A further approach to be considered in the context of statistical disambiguation is the approach to word-sense disambiguation presented in Resnik (1997). The connection of monolingual word sense disambiguation to the task of target-language disambiguation can be made by viewing candidate translations of a given lexical item as the different "senses" expressed in terms of the target language. Resnik (1997) cited an average precision of 44.3 % for his experiments on word-sense disambiguation evaluated on verb-object pairs for a random baseline of 28.5 %. His model can be interpreted as a robust, shallow, domain-unspecific, and accurate SD, but it uses a fixed semantic network which is not available in most languages. Thus, in terms of the above classification, it's main drawback is the lack of portability.

To sum up, either the suggested SDs are not robust and economic (Dagan and Ito) or they have a low accuracy even in the domain-specific case (Melamed) or they lack portability (Resnik).

In the following, we will present a new approach to SD which uses the above described latent class models as probabilistic disambiguation device. We will show that this approach yields a robust, economic SD device that is portable and has a high accuracy even in the domain-unspecific case.

5.6.2 Statistical Disambiguation

This section is particularly concerned with the use of a statistical disambiguation component that specifies the translation mapping of ambiguous nouns. This restriction to noun ambiguity can motivated by observations in Buschbeck-Wolf (1997) who states that in contrast to verbal and modifier predicates the disambiguation of nouns is difficult since it is impossible to symbolically fix all contexts in which the one or the other translation is preferred. Thus, for this task, it seems reasonable to rely on statistical information.

Consider for example the German sentence in Fig. 5.20 which is to be translated into English. Suppose that the transfer-module will be presented with an input encoding the noun argument "Karte" to be a direct object of the verbal predicate "spielen".

GER	Sie spielen ausserdem die Karte der britischen Regierung
ID-76627	You are playing the card UK Government
ENG	You are playing the UK Government card

Figure 5.20: German-English transfer

Furthermore, the symbolic components are supposed to include a word-to-word translation module translating "spielen" to "play" and presenting the possibilities "card", "chart", "map", and "ticket" as translations of "Karte".

The task of the statistical disambiguator is to specify which of the presented four English nouns should be taken as the proper argument of the predicate "spielen". The idea employed in our statistical approach is very simple: In order to decide which of the four verb-noun combinations is the best one, we take the pair that is assigned the highest joint probability by a reasonable probability model on such predicate-argument relations. For the above example, the joint probabilities $p_{LC}(\text{play}, \text{card})$, $p_{LC}(\text{play}, \text{chart})$, $p_{LC}(\text{play}, \text{map})$, $p_{LC}(\text{play}, \text{ticket})$ have to be compared. The verb-noun pair with the highest probability will then be taken as the output of the statistical disambiguator. In case there is no unique maximum in this comparison, the system yields the output "don't-know".

5.6.3 Disambiguation Experiments

The following experiments were performed with the English clustering models described in Sect. 5.3.

Evaluation on a Pseudo-Disambiguation Task

We evaluated our clustering models on a pseudo-disambiguation task similar to the one described in Sect 5.5, but specified to noun ambiguity. The task is to judge which of two nouns

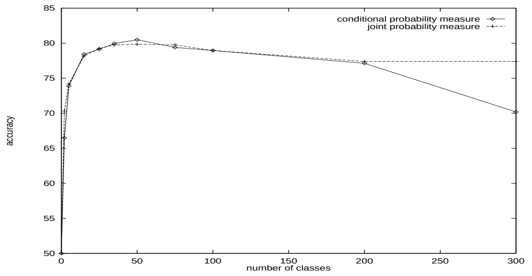


Figure 5.21: Evaluation on pseudo-disambiguation task

n and n' is more likely to appear as argument of a given verb v where the pair (v,n) has been cut out of the original training data and the pair (v,n') is constructed by pairing v with a randomly chosen noun n'. The data were built by constructing an evaluation corpus of (v,n,n') triples from a test corpus of 3,000 (v,n) pairs which was randomly cut out of the original corpus of 1,280,715 tokens. Each verb n in the test corpus was combined with a noun n' which was randomly chosen according to its frequency such that the pair (v,n') did not appear in the training and in the test corpus. Again, the elements v, n', and n had to be part of the reduced training corpus. As above, we restricted the verbs and nouns in the evaluation corpus to the ones which occurred with at least 30 and at most 3000 partners in the original training corpus. The reduced training corpus of 1,178,698 tokens was used to train a sequence of clustering models which were evaluated on the resulting 1,685 evaluation triples.

Clustering models were parametrized in starting values of EM-training, in the number of classes of the model (up to 300), and in the number of iteration steps (up to 50), resulting in a sequence of $3 \times 10 \times 6$ models. Accuracy was calculated as the number of times the model decided for $p(v,n) \geq p(v,n')$ (joint probability measure) resp. $p(v|n) \geq p(v|n')$ (conditional probability measure) out of all choices made. As shown in Fig. 5.21, we obtained an accuracy of ca. 79 % for models between 35 and 100 classes, averaged over different starting values. For models with more than 100 classes we see a small but stable overfitting effect.

Evaluation on a Smoothing Task

Recall the experiments on the smoothing power of LC models as described in Sect. 5.4. There we reported a proportion of the number of types in the English training corpus to the $V \times N$-space of 0.14 % without clustering compared to circa 93 % for clustering models with 50 classes and 50 iterations. As will be shown below, this remarkable increase in smoothing power is one of the key features of our approach to cluster-based SD.

Evaluation on a Golden Standard

To evaluate the output of the statistical disambiguator, we prepared an evaluation corpus from the sentence-aligned debates of the European parliament (mlcc = multilingual corpus for cooperation). Each language is represented by ca. 9 million tokens. The direction of translation was German to English.

angriff	aggression, assault, offence, offense, onset, onslaught, attack , charge, offensive, raid, whammy, inroad
art	fashion, fit, kind, wise, manner, type, species, mode, sort, variety, form, way
aufgabe	abandonment, task, exercise, lesson, giveup, job , office, problem, tax
auswahl	choose, eligibility, selection, choice, choosing, varity, assortment, extract, range, sample
begriff	concept, item, notion, idea
boden	floor, soil, bottom, ground, land
einrichtung	arrangement, constitution, establishment, feature, installation, institution, construction, setup, adjustment, composition, organization
erweiterung	amplification, enhancement, expansion, extension, extention, upgrade, dilatation, dilation, upgrading, add-on, increment
fehler	blemish, blunder, bug, defect, demerit, error, fail, failure, fault, flaw, mistake, shortcom, shortcoming, trouble, slip, blooper, lapse, lapsus
genehmigung	permission, approval, consent, acceptance, approbation, authorization
geschichte	history, story, tale, saga, strip
gesellschaft	companion, companionship, society, company, party, associate
grenze	boundary, frontier, limit, border, periphery, borderline, edge
grund	base, cause, ground, master, matter, reason, bottom root
karte	card, map, ticket, chart
lage	site, situation, position, bearing, layer, tier
mangel	deficiency, fault, lack, privation, scarcity, want, shortage, shortcoming, absence, dearth, demerit, desideration, desideratum, insufficiency, paucity, scarceness
menge	crowd, lot, mass, multitude, plenty, quantity, quantum , quiverful, volume, abundance, amount, aplenty, assemblage , batch, crop, deal, heap, lashings, scores, set, loads, bulk
pruefung	examination, ordeal, scrutiny, test, trial, inspection, exam, testing, tryout, verification, assay, canvass, check, checkup, inquiry, perusal, reconsideration, scruting, exa
schwierigkeit	difficulty, problem, severity, trouble, ardousness, heaviness
seite	page, side, point, aspect, party
sicherheit	certainty, certitude, immunity, safety, security , collateral , secureness, doubtlessness, sureness, guarantee, guaranty, deposit
stimme	voice, vote, tones
termin	date, appointment, meeting, time, term, deadline
verbindung	chain, conjunction, connection, connexion, fusion, incorporation, interconnection, joint , junction, link, compound, alliance , catenation, tie, union, bond, chaining, association, interface, join, liaison, contact, linkage, liaise, touch, relation
verbot	ban, interdiction, prohibition, forbade, forbad, forbiddance
verpflichtung	commitment, committal, duty, obligation, indebtedness , onus, bond, debt, duty, engagement, liability, undertaking
vertrauen	confidence, faith, reliance, trust, confidentialness, trustfulness, assurance, dependence, private, secret
wahl	choice , election, option, ballot, electoral, alternative, poll list
weg	lane, road, way, alley, route, path
widerstand	resistance, resistor, opposition, drag, resistivity
zeichen	char, character, icon, sign, signal, symbol, mark, token, figure, omen
ziel	aim, designation, destination, target, end, goal, object, objective, sightings, intention, prompt, ends
zusammenhang	coherence, context, contiguity, connection
zustimmung	accordance, agreement, approbation, consent, affirmation, allowance, approval, assent, compliance, compliancy, acclamation

Figure 5.22: 35 word dictionary extracted from online-resources

The gold standard was prepared in the following way. We gathered word-to-word translations by online-available dictionaries and eliminated German nouns for which we could not find at least two English translations showing a genuine "semantic" ambiguity. The resulting dictionary is shown in Fig. 5.22. It includes 35 German nouns with an average of 9-10 translations, i.e., in sum 333 English nouns. Based on this dictionary, we extracted all bilingual sentence pairs from the corpus which included a German noun from this dictionary in

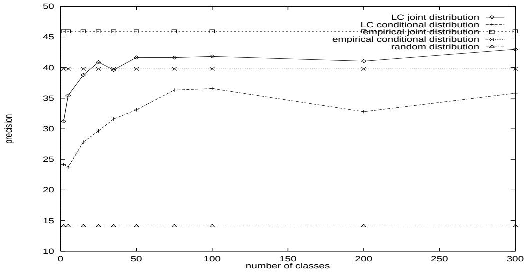


Figure 5.23: Precision of statistical disambiguation versus empirical and random choice

a verb-object position (".aso:o"). Furthermore, the English translation of the object was required to be included in our dictionary and had to appear in a similar verb-object position as the source-object for an acceptable English translation of the German verb. We marked the German noun n_g in the source-sentence, its English translation n_e as appearing in the corpus, and the English lexical verb v_e. This semi-automatic procedure resulted in a test corpus of 1341 sentences.

The goal of the statistical disambiguation was to determine for each marked German noun n_g the dictionary-specified translation $n \in Trans(n_g)$ which resulted in the most probable combination (v_e, n). That is, as a translation of n_g in the context of v_e,

$$n_e = \underset{n \in Trans(n_g)}{\arg\max} \; p_{LC}(v_e, n)$$

is selected if the arg max is defined. Otherwise, the output of the disambiguator is "don't know". The results of our LC-disambiguation tested against the gold standard are shown in Fig. 5.23 (precision) and Fig. 5.24 (recall). The clustering-based statistical disambiguator is compared with the empirical distribution of (v, n)-pairs in the training corpus and a random distribution on (v, n)-pairs. Precision measures the number of times the disambiguator under consideration chooses the same English translation as the human translator defining the gold standard. That is, precision is the number of "correct" translations according to the gold standard divided by the number of "correct" + "incorrect" translation. Recall specifies the number of times the disambiguation component chooses the "correct" translation out of the "correct" + "incorrect" + "don't know" decisions.

Fig. 5.23 shows the precision results for LC-disambiguators using a joint or a conditional probability measure, compared to the joint and conditional empirical distribution and a random distribution. The best result is obtained for the joint empirical distribution of (v, n)-pairs (45.928 %). According to the maximum likelihood paradigm, the LC-disambiguator approaches the empirical distribution in the limit of the number of classes, i.e., we see an im-

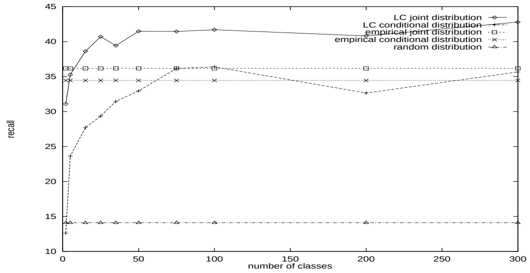


Figure 5.24: Recall of statistical disambiguation versus empirical and random choice

provement from ca. 31 % for LC-models with joint probability measure and 2 classes up to ca. 43 % for LC-models with 300 classes measured with a joint probability measure. The results are worse for both the conditional empirical distribution and the conditional LC-model, but worst for the random distribution (ca. 14 %). Note that these numbers have to be considered in the context of an average of ca. 10 translation choices for each noun.

However, as can be seen in Fig. 5.24, showing the recall results for the disambiguators, the main advantage of our LC-disambiguator is the gain in recall it has over the empirical distribution. That is, the high smoothing power of the LC-model enables a disambiguation decision for nearly every test item, i.e., $\arg\max_{n \in Trans(n_g)} p_{LC}(v_e, n)$ is never undefined since there is no case where $p_{LC}(v_e, n) = 0$ for all $n \in Trans(n_g)$. Thus we get a recall percentage of ca. 43 % for the best LC-disambiguators compared to 36.167 % for the joint empirical distribution. Random choice gives a result of ca. 14 % recall.

Note that the curves for the LC-disambiguator have the same shape as the curves resulting from the pseudo-disambiguation task reported above, thus making it a good guess to choose a proper LC-model in terms of class-cardinality from the cheaper pseudo-disambiguation task rather than choosing it from the labor-intensive evaluation on a gold standard.

In a subset of 100 test items, a human judge having access only to v_e and the set of candidates for n_e, i.e. the information used by the model, selected among translations. On this set, human performance was 39 % precision and recall. Performance for class models of size 100, 200, and 300 was 35 %, 39 %, and 45 % respectively (precision = recall). Joint empirical performance was 43 % precision and 34 % recall.

5.6.4 Discussion

We have proposed a statistical disambiguation component based on latent class models on pairs of grammatically related lexical items.

We believe the method meets the main conditions on a powerful statistical disambiguation component because:

1. The LC disambiguator has a high smoothing power, thus it is robust.
2. It resolves translation ambiguities in just one run, so it is quick.
3. It uses only minimal informations from the symbolic transfer component, so it can be called economic.
4. It relies only on monolingual information, so it is portable.
5. Evaluation on a pseudo-disambiguation task (80 % accuracy) and on a golden standard method (43 % accuracy on ca. 10 alternatives in average) shows that the method is accurate.

Bibliography

Abney, S. (1991). Parsing by chunks. In D. Bouchard and K. Leffel (Eds.), *Views on Phrase Structure*. Dortrecht: Kluwer Academic Publishers.

Abney, S. (1995). Chunks and dependencies: Bringing processing evidence to bear on syntax. In *Computational Linguistics and the Foundations of Linguistic Theory*. Stanford, CA: CSLI.

Abney, S. (1996). Chunk stylebook. Technical report, SfS, Universität Tübingen.

Baker, J. (1979). Trainable grammars for speech recognition. In D. Klatt and J. Wolf (Eds.), *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pp. 547–550.

Baum, L. E., T. Petrie, G. Soules, and N. Weiss (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics 41*(1), 164–171.

Baum, L. E. and G. A. Sell (1968). Growth transformations for functions on manifolds. *Pacific Journal of Mathematics 27*(2), 211–227.

Beckwith, R., C. Fellbaum, D. Gross, and G. A. Miller (1991). Wordnet: A lexical database organized on psycholinguistic principles. In *Lexical Acquisition – Exploiting On-Line Resources to Build a Lexicon*, Chapter 9, pp. 211–232.

Beil, F., G. Carroll, D. Prescher, S. Riezler, and M. Rooth (1998). Inside-outside estimation of a lexicalized PCFG for German. –Gold–. In *Inducing Lexicons with the EM Algorithm*, AIMS Report 4(3). IMS, Universität Stuttgart.

Booth, T. L. and R. A. Thompson (1973). Applying probability measures to abstract languages. *IEEE Transactions on Computers C-22*(5), 442–450.

Brown, P., P. deSouza, R. Mercer, V. D. Pietra, and J. Lai (1992). Class-based n-gram models of natural language. *Computational Linguistics 18*(4), 467–479.

Buschbeck-Wolf, B. (1997). Resolution on demand. Technical Report 196, Verbmobil.

Carroll, G. (1997a). *Manual pages for* `charge`*,* `hyparCharge`*, and* `tau`. IMS, Universität Stuttgart.

Carroll, G. (1997b). *Manual pages for* `supar`*,* `ultra`*,* `hypar`*, and* `genDists`. IMS, Universität Stuttgart.

Carroll, G. and M. Rooth (1998). Valence induction with a head-lexicalized PCFG. In *Proceedings of EMNLP-3*, Granada.

Charniak, E. (1995). Parsing with context-free grammars and word statistics. Technical Report CS-95-28, Department of Computer Science, Brown University.

Chomsky, N. (1965). *Aspects of the Theory of Syntax*. Cambridge, MA: M.I.T. Press.

Church, K. W., W. A. Gale, P. Hanks, and D. Hindle (1991). Using statistics in lexical analysis. In U. Zernik (Ed.), *Lexical acquisition: exploiting on-line resources to build a lexicon.*

Dagan, I. and A. Itai (1994). Word sense disambiguation using a second language monolingual corpus. *Computational Linguistics 20*, 563–596.

Dagan, I., L. Lee, and F. Pereira (1998). Similarity-based models of word cooccurrence probabilities. To appear in *Machine Learning.*

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

Duda, R. O. and P. E. Hart (1973). *Pattern classification and scene analysis.* New York: Wiley.

Ersan, M. and E. Charniak (1995). A statistical syntactic disambiguation program and what it learns. Technical Report CS-95-29, Department of Computer Science, Brown University.

Fodor, J. A. (1998). *Concepts: Where Cognitive Science Went Wrong.* Oxford: Oxford Cognitive Science Series.

Gazdar, G., E. Klein, G. K. Pullum, and I. A. Sag (1985). *Generalized Phrase Structure Grammar.* Cambridge, MA: Harvard University Press.

Grimshaw, J. (1990). *Argument Structure.* Cambridge, MA: MIT Press.

Hale, K. and S. Keyser (1993). Argument structure and the lexical expression of syntactic relations. In K. Hale and S. Keyser (Eds.), *The View from Building 20.* Cambridge, MA: MIT Press.

Hindle, D. (1983). User manual for fidditch, a deterministic parser. Technical Memorandum 7590-142, Naval Research Laboratory.

Hindle, D. (1994). A parser for text corpora. In B. Atkins and A. Zampolli (Eds.), *Computational Approaches to the Lexicon.* Oxford: Oxford University Press.

Hindle, D. and M. Rooth (1993). Structural ambiguity and lexical relations. *Computational Linguistics 19*, 103–120.

Hofmann, T. and J. Puzicha (1998). Unsupervised learning from dyadic data. Technical Report TR-98-042, International Computer Science Insitute, Berkeley, CA.

Hughes, J. (1994). *Automatically Acquiring Classification of Words.* Ph. D. thesis, University of Leeds, School of Computer Studies.

Jackendoff, R. (1977). *$\bar{X}$ Syntax: A Study in Phrase Structure.* Cambridge, MA: MIT Press.

Karttunen, L., T. Yampol, and G. Grefenstette (1994). *INFL Morphological Analyzer/Generator 3.2.9 (3.6.4).* Xerox Corporation.

Katz, S. M. (1980). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing 35.*

Klavans, J. L. and M.-Y. Kan (1998, August). The role of verbs in document analysis. In *Proceedings of the 17th International Conference on Computational Linguistics (COLING-ACL '98)*, Montreal, Canada.

Kullback, S. and R. A. Leibler (1951). On Information and Sufficiency. *Annals of Mathematical Statistics 22*, 79–86.

Kural, M. (1996). *Verb Incorporation and Elementary Predicates.* Ph. D. thesis, University of California, Los Angeles.

Lang, B. (1989). A uniform formal framework for parsing. In *Proceedings of the 1st International Workshop on Parsing Technologies (IWPT 1)*, Pittsburgh, PA, pp. 28–42.

Lang, S. (1993). *Algebra* (Third edition ed.). Reading, MA: Addison-Wesley.

Levin, B. (1993). *English Verb Classes and Alternations. A Preliminary Investigation.* Chicago/London: The University of Chicago Press.

Liberman, M. (1992). *ACL-DCI CD ROM 1.* Association for Computational Linguistics.

McLachlan, G. J. and T. Krishnan (1997). *The EM Algorithm and Extensions.* New York: Wiley.

Melamed, I. D. (1997). Word-to-word models of translational equivalence. Technical Report IRCS, Nr. 98-08, University of Pennsylvania.

Ney, H., U. Essen, and R. Kneser (1994). On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language 8*, 1–38.

Pereira, F., N. Tishby, and L. Lee (1993). Distributional clustering of English words. In *Proceedings of the 31th Annual Meeting of the ACL*, Columbus, Ohio.

Pollard, C. and I. A. Sag (1987). *Information-Based Syntax and Semantics, Vol. 1: Fundamentals.* Stanford, CA: CSLI.

Resnik, P. (1993). *Selection and information: A class-based approach to lexical relationships.* Ph. D. thesis, University of Pennsylvania, CIS Department.

Resnik, P. (1997). Selectional preference and sense disambiguation. Paper presented at: Tagging Text with Lexical Semantics: Why, What, and How?, Workshop in connection with ANLP 97, Washington, D.C.

Ribas, F. (1994). An experiment on learning appropriate selectional restrictions from a parsed corpus. In *Proceedings of COLING-94*, Kyoto, Japan.

Ribas, F. (1995). On learning more appropriate selectional restrictions. In *Proceedings of the 7th Conference of the EACL*, Dublin, Ireland.

Rooth, M. (1998). Two-dimensional clusters in grammatical relations. In *Inducing Lexicons with the EM Algorithm*, AIMS Report 4(3). Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Rooth, M. (Ms.). Two-dimensional clusters in grammatical relations. In *Symposium on Representation and Acquisition of Lexical Knowledge: Polysemy, Ambiguity, and Generativity*, AAAI 1995 Spring Symposium Series. Stanford University.

Rooth, M., S. Riezler, D. Prescher, G. Carroll, and F. Beil (1998). EM-based clustering for NLP applications. In *Inducing Lexicons with the EM Algorithm*, AIMS Report 4(3). Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Saul, L. K. and F. Pereira (1997). Aggregate and mixed-order Markov models for statistical language processing. In *Proceedings of EMNLP-2.*

Schiller, A. and C. Stöckert (1995). *DMOR.* IMS, Universität Stuttgart.

Schmid, H. (1999). *Manual page for* `lopar`. IMS, Universität Stuttgart.

Schuhmacher, H. (1986). *Verben in Feldern. Valenzwörterbuch zur Syntax und Semantik deutscher Verben.* Berlin: de Gruyter.

Schulte im Walde, S. (1998). Automatic semantic classification of verbs according to their alternation behaviour. Master's thesis, Institut für maschinelle Sprachverarbeitung, Universität Stuttgart.

Skut, W. and T. Brants (1998). A maximum-entropy partial parser for unrestricted text. In *Proceedings of the Sixth Workshop on Very Large Corpora*, Montréal, Québec.

Stowell, T. (1981). *Origins of Phrase Structure.* Ph. D. thesis, MIT.

Wahba, G. and S. Wold (1975). A completely automatic french curve: Fitting spline functions by cross validation. *Commun. Statist. 4*, 1–17.