

Stochastic Attribute Value Grammars

Rob Malouf and Miles Osborne

ESSLLI'01

August, 2001
Helsinki, Finland

Stochastic Attribute Value Grammars

Rob Malouf and Miles Osborne

European Summer School in Logic, Language and Information

20–24 August 2001

Unification-based attribute-value grammar formalisms such as Lexical-Functional Grammar and Head-Driven Phrase Structure Grammar have proven to be highly successful for practical large-scale grammar development. However, realistic applications of attribute-value grammars for natural language parsing or generation require the use of sophisticated statistical techniques for resolving ambiguities. This one-week course will provide an introduction to the maximum entropy principle and the construction of maximum entropy models for natural language processing. Through a combination of lectures and, as local computing facilities permit, hands-on lab exercises, students will investigate the implementation of maximum entropy models for attribute-value grammars, including such topics as ambiguity identification, feature selection, and model training and evaluation.

This course will assume a basic knowledge of probability theory, and some experience in grammar development or programming in a high level language would be helpful.

Readings

- Abney, Steven. 1996. “Statistical methods and linguistics.” In Judith Klavans and Philip Resnik (eds.), *The Balancing Act*. The MIT Press, Cambridge, MA. pp. 1–26.
- Abney, Steven. 1997. “Stochastic attribute-value grammars.” *Computational Linguistics* **23**:597–618.
- Berger, Adam, Stephen Della Pietra, and Vincent Della Pietra. 1996. “A maximum entropy approach to natural language processing.” *Computational Linguistics* **22**:39–71.
- Bouma, Gosse, Gertjan van Noord, and Robert Malouf. In press. “Alpino: Wide coverage computational analysis of Dutch.” In *Computational Linguistics in the Netherlands*.
- Della Pietra, Stephen, Vincent Della Pietra, and John Lafferty. 1997. “Inducing features of random fields.” *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**:380–393.
- Johnson, Mark, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. “Estimators for stochastic “unification-based” grammars.” In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL 1999)*. pp. 535–541.
- Osborne, Miles. 2000. “Estimation of stochastic attribute-value grammars using an informative sample.” In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*.
- Pollard, Carl. 1997. *Lectures on the Foundations of HPSG*. Unpublished manuscript: Ohio State University.

STATISTICAL METHODS AND LINGUISTICS

Steven Abney
University of Tübingen

In: Judith Klavans and Philip Resnik, eds., *The Balancing Act*. MIT Press,
Cambridge, MA. 1996.

abney@sfs.nphil.uni-tuebingen.de
<http://www.sfs.nphil.uni-tuebingen.de/~abney/>
Wilhelmstr. 113, 72074 Tübingen, Germany

STATISTICAL METHODS AND LINGUISTICS

Steven Abney
University of Tübingen

In the space of the last ten years, statistical methods have gone from being virtually unknown in computational linguistics to being a fundamental given. In 1996, no one can profess to be a computational linguist without a passing knowledge of statistical methods. HMM's are as de rigeur as LR tables, and anyone who cannot at least use the terminology persuasively risks being mistaken for kitchen help at the ACL banquet.

More seriously, statistical techniques have brought significant advances in broad-coverage language processing. Statistical methods have made real progress possible on a number of issues that had previously stymied attempts to liberate systems from toy domains; issues that include disambiguation, error correction, and the induction of the sheer volume of information requisite for handling unrestricted text. And the sense of progress has generated a great deal of enthusiasm for statistical methods in computational linguistics.

However, this enthusiasm has not been catching in linguistics proper. It is always dangerous to generalize about linguists, but I think it is fair to say that most linguists are either unaware (and unconcerned) about trends in computational linguistics, or hostile to current developments. The gulf in basic assumptions is simply too wide, with the result that research on the other side can only seem naive, ill-conceived, and a complete waste of time and money.

In part the difference is a difference of goals. A large part of computational linguistics focuses on practical applications, and is little concerned with human language processing. Nonetheless, at least some computational linguists aim to advance our scientific understanding of the human language faculty by better understanding the computational properties of language. One of the most interesting and challenging questions about human language computation is just how people are able to deal so effortlessly with the very issues that make processing unrestricted text so difficult. Statistical methods provide the most promising current answers, and as a result the excitement about statistical methods is also shared by those in the cognitive reaches of computational linguistics.

In this paper, I would like to communicate some of that excitement to fellow linguists, or at least, perhaps, to make it comprehensible. There is no denying that there is a culture clash between theoretical and computational linguistics that serves to reinforce mutual prejudices. In charicature, computational linguists believe that by throwing more cycles and more raw text into their statistical black box, they can dispense with linguists altogether, along with their fanciful Rube Goldberg theories about exotic linguistic phenomena. The

I would like to thank Tilman Hoehle, Graham Katz, Marc Light, and Wolfgang Sternefeld for their comments on an earlier draft of this paper. All errors and outrageous opinions are of course my own.

linguist objects that, even if those black boxes make you oodles of money on speech recognizers and machine-translation programs (which they don't), they fail to advance our understanding. I will try to explain how statistical methods just might contribute to understanding of the sort that linguists are after.

This paper, then, is essentially an apology, in the old sense of *apology*. I wish to explain why we would do such a thing as to use statistical methods, and why they are not really such a bad thing, maybe not even for linguistics proper.

1 Language Acquisition, Language Variation, and Language Change

I think the most compelling, though least well-developed, arguments for statistical methods in linguistics come from the areas of language acquisition, language variation, and language change.

Language acquisition. Under standard assumptions about the grammar, we would expect the course of language development to be characterized by abrupt changes, each time the child learns or alters a rule or parameter of the grammar. If, as seems to be the case, changes in child grammar are actually reflected in changes in relative frequencies of structures that extend over months or more, it is hard to avoid the conclusion that the child has a probabilistic or weighted grammar in some form. The form that would perhaps be least offensive to mainstream sensibilities is a grammar in which the child “tries out” rules for a time. During the trial period, both the new and old versions of a rule co-exist, and the probability of using one or the other changes with time, until the probability of using the old rule finally drops to zero. At any given point, in this picture, a child's grammar is a stochastic (i.e., probabilistic) grammar.

An aspect of this little illustration that bears emphasizing is that the probabilities are added to a grammar of the usual sort. A large part of what is meant by “statistical methods” in computational linguistics is the study of stochastic grammars of this form: grammars obtained by adding probabilities in a fairly transparent way to “algebraic” (i.e., non-probabilistic) grammars. Stochastic grammars of this sort do not constitute a rejection of the underlying algebraic grammars, but a supplementation. This is quite different from some uses to which statistical models (most prominently, neural networks) are put, in which attempts are made to model some approximation of linguistic behavior with an undifferentiated network, with the result that it is difficult or impossible to relate the network's behavior to a linguistic understanding of the sort embodied in an algebraic grammar. (It should, however, be pointed out that the problem with such applications does not lie with neural nets, but with the unenlightening way they are put to use.)

Language change. Similar comments apply, on a larger scale, to language change. If the units of change are as algebraic grammars lead us to expect—rules or pa-

rameters or the like—we would expect abrupt changes. We might expect some poor bloke to go down to the local pub one evening, order “Ale!”, and be served an eel instead, because the Great Vowel Shift happened to him a day too early.¹ In fact, linguistic changes that are attributed to rule changes or changes of parameter settings take place gradually, over considerable stretches of time, measured in decades or centuries. It is more realistic to assume that the language of a speech community is a stochastic composite of the languages of the individual speakers, described by a stochastic grammar. In the stochastic “community” grammar, the probability of a given construction reflects the relative proportion of speakers who use the construction in question. Language change consists in shifts in relative frequency of constructions (rules, parameter settings, etc.) in the community. If we think of speech communities as populations of grammars that vary within certain bounds, and if we think of language change as involving gradual shifts in the center of balance of the grammar population, then statistical models are of immediate applicability [25].

In this picture, we might still continue to assume that an adult monolingual speaker possesses a particular algebraic grammar, and that stochastic grammars are only relevant for the description of communities of varying grammars. However, we must at least make allowance for the fact that individuals routinely comprehend the language of their community, with all its variance. This rather suggests that at least the grammar used in language comprehension is stochastic. I return to this issue below.

Language variation. There are two senses of language variation I have in mind here: dialectology, on the one hand, and typology, on the other. It is clear that some languages consist of a collection of dialects that blend smoothly one into the other, to the point that the dialects are more or less arbitrary points in a continuum. For example, Tait describes Inuit as “a fairly unbroken chain of dialects, with mutual intelligibility limited to proximity of contact, the furthest extremes of the continuum being unintelligible to each other” [26, p.3]. To describe the distribution of Latin American native languages, Kaufman defines a *language complex* as “a geographically continuous zone that contains linguistic diversity greater than that found within a single language . . . , but where internal linguistic boundaries similar to those that separate clearly discrete languages are lacking” [14, p.31]. The continuousness of changes with geographic distance is consistent with the picture of a speech community with grammatical variance, as sketched above. With geographic distance, the mix of frequency of usage of various constructions changes, and a stochastic grammar of some sort is an appropriate model [15].

Similar comments apply in the area of typology, with a twist. Many of the universals of language that have been identified are statistical rather than abso-

¹I have read this anecdote somewhere before, but have been unable to find the citation. My apologies to the unknown author.

lute, including rough statements about the probability distribution of language features (“head-initial and head-final languages are about equally frequent”) or conditional probability distributions (“postpositions in verb-initial languages are more common than prepositions in verb-final languages”) [11, 12]. There is as yet no model of how this probability distribution comes about, that is, how it arises from the statistical properties of language change. Which aspects of the distribution are stable, and which would be different if we took a sample of the world’s languages 10,000 years ago or 10,000 years hence? There is now a vast body of mathematical work on stochastic processes and the dynamics of complex systems (which includes, but is not exhausted by, work on neural nets), much of which is of immediate relevance to these questions.

In short, it is plausible to think of all of these issues—language acquisition, language change, and language variation—in terms of populations of grammars, whether those populations consist of grammars of different speakers or sets of hypotheses a language learner entertains. When we examine populations of grammars varying within bounds, it is natural to expect statistical models to provide useful tools.

2 Adult Monolingual Speakers

But what about an adult monolingual speaker? Ever since Chomsky, linguistics has been firmly committed to the idealization to an adult monolingual speaker in a homogeneous speech community. Do statistical models have anything to say about language under that idealization?

In a narrow sense, I think the answer is probably not. Statistical methods bear mostly on all the issues that are outside the scope of interest of current mainstream linguistics. In a broader sense, though, I think that says more about the narrowness of the current scope of interest than about the linguistic importance of statistical methods. Statistical methods are of great linguistic interest because the issues they bear on are linguistic issues, and essential to an understanding of what human language is and what makes it tick. We must not forget that the idealizations that Chomsky made were an expedient, a way of managing the vastness of our ignorance. One aspect of language is its algebraic properties, but that is only one aspect of language, and certainly not the only important aspect. Also important are the statistical properties of language communities. And stochastic models are also essential for understanding language production and comprehension, particularly in the presence of variation and noise. (I will focus here on comprehension, though considerations of language production have also provided an important impetus for statistical methods in computational linguistics [22, 23].)

To a significant degree, I think linguistics has lost sight of its original goal, and turned Chomsky’s expedient into an end in itself. Current theoretical syntax gives a systematic account of a very narrow class of data, judgments about

the well-formedness of sentences for which the intended structure is specified, where the judgments are adjusted to eliminate gradations of goodness and other complications. Linguistic data other than structure judgments are classified as “performance” data, and the adjustments that are performed on structure-judgment data are deemed to be corrections for “performance effects”. Performance is considered the domain of psychologists, or at least, not of concern to linguistics.

The term *performance* suggests that the things that the standard theory abstracts away from or ignores are a natural class; they are data that bear on language processing but not language structure. But in fact a good deal that is labelled “performance” is not computational in any essential way. It is more accurate to consider performance to be negatively defined: it is whatever the grammar does not account for. It includes genuinely computational issues, but a good deal more that is not. One issue I would like to discuss in some detail is the issue of grammaticality and ambiguity judgments about sentences as opposed to structures. These judgments are no more or less computational than judgments about structures, but it is difficult to give a good account of them with grammars of the usual sort; they seem to call for stochastic, or at least weighted, grammars.

2.1 Grammaticality and ambiguity

Consider the following:

- (1) a. the a are of I
- b. the cows are grazing in the meadow
- c. John saw Mary

The question is the status of these examples with respect to grammaticality and ambiguity. The judgments here, I think, are crystal clear: (1a) is word salad, and (1b) and (c) are unambiguous sentences.

In point of fact, (1a) is a grammatical noun phrase, and (1b) and (c) are at least two ways ambiguous, the non-obvious reading being as a noun phrase. Consider: an *are* is a measure of area, as in *a hectare is a hundred ares*, and letters of the alphabet may be used as nouns in English (“*Written on the sheet was a single lowercase a,*” “*As described in section 2 paragraph b . . .*”). Thus (1a) has a structure in which *are* and *I* are head nouns, and *a* is a modifier of *are*. This analysis even becomes perfectly natural in the following scenario. Imagine we are surveyors, and that we have mapped out a piece of land into large segments, designated with capital letters, and subdivided into one-are sub-segments, designated with lower-case letters. Then *the a are of I* is a perfectly natural description for a particular parcel on our map.

As for (1b), *are* is again the head noun, *cows* is a premodifier, and *grazing in the meadow* is a postmodifier. It might be objected that plural nouns cannot

be nominal premodifiers, but in fact they often are: consider *the bonds market, a securities exchange, he is vice president and media director, an in-home health care services provider, Hartford's claims division, the financial-services industry, its line of systems management software*. (Several of these examples are extracted from the Wall Street Journal.)

It may seem that examples (1a) and (b) are illustrative only of a trivial and artificial problem that arises because of a rare usage of a common word. But the problem is not trivial: without an account of ‘rare usage’, we have no way of distinguishing between genuine ambiguities and these spurious ambiguities. Alternatively, one might object that if one does not know that *are* has a reading as a noun, then *are* is actually unambiguous in one’s idiolect, and (1a) is genuinely ungrammatical. But in that case the question becomes why *a hectare is a hundred ares* is not judged equally ungrammatical by speakers of the idiolect in question.

Further, (1c) illustrates that the rare usage is not an essential feature of examples (a) and (b). *Saw* has a reading as a noun, which may be less frequent than the verb reading, but is hardly a rare usage. Proper nouns can modify (*Gatling gun*) and be modified by (*Typhoid Mary*) common nouns. Hence, *John saw Mary* has a reading as a noun phrase, referring to the Mary who is associated with a kind of saw called a John saw.

It may be objected that constructions like *Gatling gun* and *Typhoid Mary* belong to the lexicon, not the grammar, but however that may be, they are completely productive. I may not know what *Cohen equations, the Russia house, or Abney sentences* are, but if not, then the denotata of *Cohen’s equations, the Russian house, or those sentences of Abney’s* are surely equally unfamiliar.² Likewise I may not know who *pegleg Pete* refers to, or *riverboat Sally*, but that does not make the constructions any less grammatical or productive.

The problem is epidemic, and it snowballs as sentences grow longer. One often hears in computational linguistics about completely unremarkable sentences with hundreds of parses, and that is in fact no exaggeration. Nor is it merely a consequence of having a poor grammar. If one examines the undesired analyses, one generally finds that they are extremely implausible, and often do considerable violence to ‘soft’ constraints like heaviness constraints or the number and sequence of modifiers, but no one piece of the structure is outright ungrammatical.

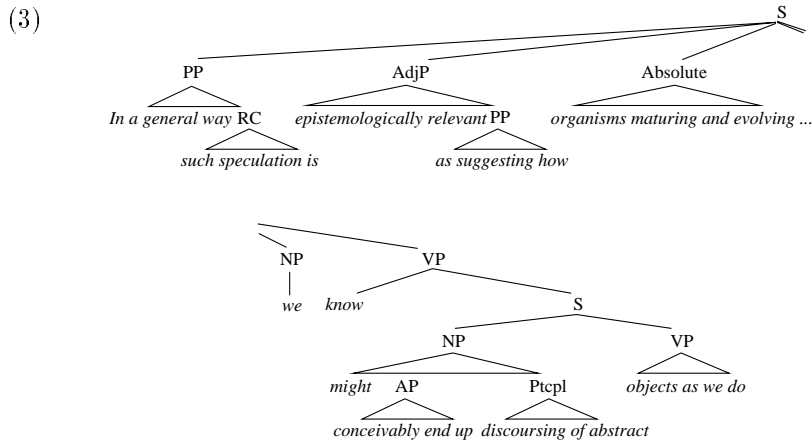
To illustrate, consider this sentence, drawn more or less at random from a book (Quine’s *Word and Object*) drawn more or less at random from my shelf:

- (2) In a general way such speculation is epistemologically relevant, as suggesting how organisms maturing and evolving in the physical environ-

²There are also syntactic grounds for doubt about the assumption that noun-noun modification belongs to the lexicon. Namely, adjectives can intervene between the modifying noun and the head noun. (Examples are given later in this section.) If adjective modification belongs to the syntax, and if there are no discontinuous words or movement of pieces of lexical items, then at least some modification of nouns by nouns must take place in the syntax.

ment we know might conceivably end up discoursing of abstract objects as we do. [28, p. 123]

One of the many spurious structures this sentence might receive is the following:



There are any number of criticisms one can direct at this structure, but I believe none of them are fatal. It might be objected that the PP-AdjP-Absolute sequence of sentential premodifiers is illegitimate, but each is individually fine, and there is no hard limit on stacking them. One can even come up with relatively good examples with all three modifiers, e.g.: [PP *on the beach*] [AdjP *naked as jaybirds*] [Absolute *waves lapping against the shore*] *the wild boys carried out their bizarre rituals*.

Another point of potential criticism is the question of licensing the elided sentence after *how*. In fact its content could either be provided from preceding context or from the rest of the sentence, as in *though as yet unable to explain how, astronomers now know that stars develop from specks of grit in giant oysters*.

Might is taken here as a noun, as in *might and right*. The AP *conceivably end up* may be a bit mysterious: *end up* is here an adjectival, as in *we turned the box end up*. *Abstract* is unusual as a mass noun, but can in fact be used as one, as for example in *the article consisted of three pages of abstract and only two pages of actual text*.

One might object that the NP headed by *might* is bad because of the multiple postmodifiers, but in fact there is no absolute constraint against stacking nominal postmodifiers, and good examples can be constructed with the same structure: *marlinespikes, business end up, sprinkled with tabasco sauce, can be a powerful deterrent against pigeons*. Even the commas are not absolutely required. The strength of preference for them depends on how heavy the modifiers are: cf. *strength judiciously applied increases the effectiveness of diplomacy, a cup*

*of peanuts unshelled in the stock adds character.*³

In short, the structure (3) seems to be best characterized as grammatical, though it violates any number of parsing preferences and is completely absurd.

One might think that one could eliminate ambiguities by turning some of the dispreferences into absolute constraints. But attempting to eliminate unwanted readings that way is like squeezing a balloon: every dispreference that is turned into an absolute constraint to eliminate undesired structures has the unfortunate side effect of eliminating the desired structure for some other sentence. No matter how difficult it is to think up a plausible example that violates the constraint, some writer has probably already thought one up by accident, and we will improperly reject his sentence as ungrammatical if we turn the dispreference into an absolute constraint. To illustrate: if a noun is premodified by both an adjective and another noun, standard grammars require the adjective to come first, inasmuch as the noun adjoins to N^0 but the adjective adjoins to \bar{N} . It is not easy to think up good examples that violate this constraint. Perhaps the reader would care to try before reading the examples in the footnote.⁴

Not only is my absurd analysis (3) arguably grammatical, there are many, many equally absurd analyses to be found. For example, *general* could be a noun (the army officer) instead of an adjective, or *evolving in* could be analyzed as a particle verb, or *the physical* could be a noun phrase (a physical exam)—not to mention various attachment ambiguities for coordination and modifiers, giving a multiplicative effect. The consequence is considerable ambiguity for a sentence that is perceived to be completely unambiguous.

Now perhaps it seems I am being perverse, and I suppose I am. But it is a perversity that is implicit in grammatical descriptions of the usual sort, and it emerges unavoidably as soon as we systematically examine the structures that the grammar assigns to sentences. Either the grammar assigns too many structures to sentences like (2), or it incorrectly predicts that examples like *three pages of abstract* or *a cup of peanuts unshelled in the stock* have no well-formed structure.

To sum up, there is a problem with grammars of the usual sort: their predictions about grammaticality and ambiguity are simply not in accord with human perceptions. The problem of how to identify the correct structure from among the in-principle possible structures provides one of the central motivations for the use of weighted grammars in computational linguistics. A weight is assigned to each aspect of structure permitted by the grammar, and the weight of a particular analysis is the combined weight of the structural features that make it up. The analysis with the greatest weight is predicted to be the perceived analysis for a given sentence.

³Cf. this passage from Tolkien: “Their clothes were mended as well as their bruises their tempers and their hopes. Their bags were filled with food and provisions light to carry but strong to bring them over the mountain passes.” [27, p.61]

⁴*Maunder climatic cycles, ice-core climatological records, a Kleene-star transitive closure, Precambrian era solar activity, highland igneous formations.*

Before describing in more detail how weighted grammars contribute to a solution to the problem, though, let me address an even more urgent issue: is this even a linguistic problem?

2.2 Is this linguistics?

Under usual assumptions, the fact that the grammar predicts grammaticality and ambiguity where none is perceived is not a linguistic problem. The usual opinion is that perception is a matter of performance, and that grammaticality alone does not predict performance; we must also include non-linguistic factors like plausibility and parsing preferences and maybe even probabilities.

Grammaticality and acceptability. The implication is that *perceptions* of grammaticality and ambiguity are not linguistic data, but performance data. This stance is a bit odd—aren't grammaticality judgments perceptions? And what do we mean by "performance data"? It would be one thing if we were talking about data that clearly has to do with the course of linguistic computation, data like response times and reading times, or regressive eye movement frequencies, or even more outlandish things like PET scans or ERP traces. But human perceptions (judgments, intuitions) about grammaticality and ambiguity are classic linguistic data. What makes the judgments concerning examples (1a-c) performance data? All linguistic data is the result of little informal psycholinguistic experiments that linguists perform on themselves, and the experimental materials are questions of the form "Can you say this?" "Does this mean this?" "Is this ambiguous?" "Are these synonymous?"

Part of the answer is that the judgments about examples (1a-c) are judgments about sentences alone rather than about sentences with specified structures. The usual sort of linguistic judgment is a judgment about the goodness of a particular structure, and example sentences are only significant as bearers of the structure in question. If any choice of words and any choice of context can be found that makes for a good sentence, the structure is deemed to be good. The basic data are judgments about structured sentences in context—that is, sentences plus a specification of the intended structure and intended context—but this basic data is used only grouped in sets of structured contextualized sentences having the same (possibly partial) structure. Such a set is defined to be good just in case any structured contextualized sentence it contains is judged to be good. Hence a great deal of linguists' time is spent in trying to find some choice of words and some context to get a clear positive judgment, in order to show that a structure of interest is good.

As a result, there is actually no intent that the grammar *predict*—that is, generate—individual structured sentence judgments. For a given structured sentence, the grammar only predicts whether there is some sentence with the same structure that is judged to be good.

For the examples (1), then, we should say that the structure

[NP the [N a] [N are] [PP of [N I]]]

is indeed grammatical in the technical sense, since it is acceptable in at least one context, and since every piece of the structure is attested in acceptable sentences.

The grouping of data by structure is not the only way that standard grammars fail to predict acceptability and ambiguity judgments. Judgments are rather smoothly graded, but goodness according to the grammar is all or nothing. Discrepancies between grammar and data are ignored if they involve sentences containing center embedding, parsing preference violations, garden path effects, or in general if their badness can be ascribed to “processing complexity”.⁵

Grammar and computation. The difference between structure judgments and string judgments is not that the former is “competence data” in some sense and the latter is “performance data”. Rather, the distinction rests on a working assumption about how the data are to be explained, namely, that the data is a result of the interaction of grammatical constraints with computational constraints. Certain aspects of the data are assumed to be reflections of grammatical constraints, and everything else is ascribed to failures of the processor to translate grammatical constraints transparently into behavior, whether because of memory limits or heuristic parsing strategies or whatever obscure mechanisms create gradedness of judgments. We are justified in ignoring those aspects of the data that we ascribe to the idiosyncracies of the processor.

But this distinction does not hold up under scrutiny. Dividing the human language capacity into grammar and processor is only a manner of speaking, a way of dividing things up for theoretical convenience. It is naive to expect the logical grammar/processor division to correspond to any meaningful physiological division—say, two physically separate neuronal assemblies, one functioning as a store of grammar rules and the other as an active device that accesses the grammar-rule store in the course of its operation. And even if we *did* believe in a physiological division between grammar and processor, we have no evidence at all to support that belief; it is not a distinction with any empirical content.

A couple of examples might clarify why I say that the grammar/processor distinction is only for theoretical convenience. Grammars and syntactic structures are used to describe computer languages as well as human languages, but typical compilers do not access grammar-rules or construct parse-trees. At

⁵In addition, there are properties of grammaticality judgments of a different sort that are not being modelled, properties that are poorly understood and somewhat worrisome. Disagreements arise not infrequently among judges—it is more often the case than not that I disagree with at least some of the judgments reported in syntax papers, and I think my experience is not unusual. Judgments seem to change with changing theoretical assumptions: a sentence that sounds “not too good” when one expects it to be bad may sound “not too bad” if a change in the grammar changes one’s expectations. And judgments change with exposure. Some constructions that sound terrible on a first exposure improve considerably with time.

the level of description of the operation of the compiler, grammar-rules and parse-trees exist only “virtually” as abstract descriptions of the course of the computation being performed. What is separately characterized as, say, grammar versus parsing strategy at the logical level is completely intermingled at the level of compiler operation.

At the other extreme, the constraints that probably have the strongest computational flavor are the parsing strategies that are considered to underlie garden-path effects. But it is equally possible to characterize parsing preferences in grammatical terms. For example, the low attachment strategy can be characterized by assigning a cost to structures of the form $[_{X^{i+1}} X^i Y Z]$ proportional to the depth of the subtree Y . The optimal structure is the one with the least cost. Nothing depends on how trees are actually computed: the characterization is only in terms of the shapes of trees.

If we wish to make a distinction between competence and computation, an appropriate distinction is between *what* is computed and *how* it is computed. By this measure, most “performance” issues are not computational issues at all. Characterizing the perceptions of grammaticality and ambiguity described in the previous section does not necessarily involve any assumptions about the computations done during sentence perception. It only involves characterizing the set of structures that are perceived as belonging to a given sentence. That can be done, for example, by defining a weighted grammar that assigns costs to trees, and specifying a constant C such that only structures whose cost is within distance C of the best structure are predicted to be perceived. How the set thus defined is actually computed during perception is left completely open.

We may think of competence versus performance in terms of knowledge versus computation, but that is merely a manner of speaking. What is really at issue is an idealization of linguistic data for the sake of simplicity.

The frictionless plane, autonomy and isolation. Appeal is often made to an analogy between competence and frictionless planes in mechanics. Syntacticians focus on the data that they believe to contain the fewest complicating factors, and “clean up” the data to remove what they believe to be remaining complications that obscure simple, general principles of language.

That is proper and laudable, but it is important not to lose sight of the original problem, and not to mistake complexity for irrelevancy. The test of whether the simple principles we think we have found actually have explanatory power is how well they fare in making sense of the larger picture. There is always the danger that the simple principles we arrive at are artifacts of our data selection and data adjustment. For example, it is sometimes remarked how marvelous it is that a biological system like language should be so discrete and clean, but in fact there is abundant gradedness and variability in the original data; the evidence for the discreteness and cleanness of language seems to be mostly evidence we ourselves have planted.

It has long been emphasized that syntax is autonomous. The doctrine is older than Chomsky; for example, Tesnière writes “...la syntaxe. Il est **autonome**” (emphasis in the original). To illustrate that structure cannot be equated with meaning, he presents the sentence pair:

le signal vert indique le voie libre
le symbole veritable impose le vitesse lissant

The similarity to Chomsky’s later but more famous pair

revolutionary new ideas appear infrequently
colorless green ideas sleep furiously

is striking.

But autonomy is not the same as isolation. Syntax is autonomous in the sense that it cannot be reduced to semantics; well-formedness is not identical to meaningfulness. But syntax in the sense of an algebraic grammar is only one piece in an account of language, and it stands or falls on how well it fits into the larger picture.

The holy grail. The larger picture, and the ultimate goal of linguistics, is to describe language in the sense of that which is produced in language production, comprehended in language comprehension, acquired in language acquisition, and, in aggregate, that which varies in language variation and changes in language change.

I have always taken the holy grail of generative linguistics to be to characterize a class of models, each of which represents a particular (potential or actual) human language L , and characterizes a speaker of L by defining the class of sentences a speaker of L produces, the structures that a speaker of L perceives for sentences; in short, by predicting the linguistic data that characterizes a speaker of L .

A “Turing test” for a generative model would be something like the following. If we use the model to generate sentences at random, the sentences that are produced are judged by humans to be clearly sentences of the language—to “sound natural”. And in the other direction, if humans judge a sentence (or non-sentence) to have a particular structure, the model should also assign precisely that structure to the sentence.

Natural languages are such that these tests cannot be passed by an unweighted grammar. An unweighted grammar distinguishes only between grammatical and ungrammatical structures, and that is not enough. “Sounding natural” is a matter of degree. What we must mean by “randomly generating natural-sounding sentences” is that sentences are weighted by the degree to which they sound natural, and we sample sentences with a probability that accords with their weight. Moreover, the structure that people assign to a sentence is the structure they judge to have been intended by the speaker, and that

judgment is also a matter of degree. It is not enough for the grammar to define the set of structures that could possibly belong to the sentence; the grammar should predict which structures humans actually perceive, and what the relative weights are in cases where humans are uncertain about which structure the speaker intended.

The long and little of it is, weighted grammars (and other species of statistical methods) characterize language in such a way as to make sense of language production, comprehension, acquisition, variation, and change. These are linguistic, and not computational issues, a fact that is obscured by labelling everything “performance” that is not accounted for by algebraic grammars. What is really at stake with “competence” is a provisional simplifying assumption, or an expression of interest in certain subproblems of linguistics. There is certainly no indicting an expression of interest, but it is important not to lose sight of the larger picture.

3 How Statistics Helps

Accepting that there are divergences between theory and data—for example, the divergence between predicted and perceived ambiguity—and accepting that this is a linguistic problem, and that it is symptomatic of the incompleteness of standard grammars, how does adding weights or probabilities help make up the difference?

Disambiguation. As already mentioned, the problem of identifying the correct parse—the parse that humans perceive—among the possible parses is a central application of stochastic grammars in computational linguistics. The problem of defining which analysis is correct is not a computational problem, however; the computational problem is describing an algorithm to compute the correct parse. There are a variety of approaches to the problem of defining the correct parse. A stochastic context-free grammar provides a simple illustration. Consider the sentence *John walks*, and the grammar

- | | | | |
|-----|----|-----------------------|-----|
| (4) | 1. | $S \rightarrow NP V$ | .7 |
| | 2. | $S \rightarrow NP$ | .3 |
| | 3. | $NP \rightarrow N$ | .8 |
| | 4. | $NP \rightarrow N N$ | .2 |
| | 5. | $N \rightarrow John$ | .6 |
| | 6. | $N \rightarrow walks$ | .4 |
| | 7. | $V \rightarrow walks$ | 1.0 |

According to grammar (4), *John walks* has two analyses, one as a sentence and one as a noun phrase. (The rule $S \rightarrow NP$ represents an utterance consisting of a single noun phrase.) The numbers in the rightmost column represent the weights of rules. The weight of an analysis is the product of the weights of the rules used

in its derivation. In the sentential analysis of *John walks*, the derivation consists of rules 1, 3, 5, 7, so the weight is $(.7)(.8)(.6)(1.0) = .336$. In the noun-phrase analysis, the rules 2, 4, 5, 6 are used, so the weight is $(.3)(.2)(.6)(.4) = .0144$. The weight for the sentential analysis is much greater, predicting that it is the one perceived. More refined predictions can be obtained by hypothesizing that an utterance is perceived as ambiguous if the next-best case is not too much worse than the best. If “not too much worse” is interpreted as a ratio of, say, not more than 2:1, we predict that *John walks* is perceived as unambiguous, as the ratio between the weights of the parses is 23:1.⁶

Degrees of grammaticality. Gradations of acceptability are not accommodated in algebraic grammars: a structure is either grammatical or not. The idea of degrees of grammaticality has been entertained from time to time, and some classes of ungrammatical structures are informally considered to be “worse” than others (most notably, ECP violations versus subadjacency violations). But such degrees of grammaticality as have been considered have not been accorded a formal place in the theory. Empirically, acceptability judgments vary widely across sentences with a given structure, depending on lexical choices and other factors. Factors that cannot be reduced to a binary grammaticality distinction are either poorly modelled or ignored in standard syntactic accounts.

Degrees of grammaticality arise as uncertainty in answering the question “Can you say X?” or perhaps more accurately, “If you said X, would you feel you had made an error?” As such, they reflect degrees of error in speech production. The null hypothesis is that the same measure of goodness is used in both speech production and speech comprehension, though it is actually an open question. At any rate, the measure of goodness that is important for speech comprehension is not degree of grammaticality alone, but a global measure that combines degrees of grammaticality with at least naturalness and structural preference (i.e., “parsing strategies”).

We must also distinguish degrees of grammaticality, and indeed, global goodness, from the probability of producing a sentence. Measures of goodness and probability are mathematically similar enhancements to algebraic grammars, but goodness alone does not determine probability. For example, for an infinite language, probability must ultimately decrease with length, though arbitrarily long sentences may be perfectly good.

Perhaps one reason that degrees of grammaticality have not found a place in standard theory is the question of where the numbers come from, if we permit continuous degrees of grammaticality. The answer to where the numbers come from is *parameter estimation*. Parameter estimation is well-understood for a

⁶The hypothesis that only the best structure (or possibly, structures) are perceptible is somewhat similar to current approaches to syntax in which grammaticality is defined as optimal satisfaction of constraints or maximal economy of derivation. But I will not hazard a guess here about whether that similarity is significant or mere happenstance.

number of models of interest, and can be seen psychologically as part of what goes on during language acquisition.

Naturalness. It is a bit difficult to say precisely what I mean by naturalness. A large component is plausibility, but not plausibility in the sense of world knowledge, but rather plausibility in the sense of selectional preferences, that is, semantic sortal preferences that predicates place on their arguments.

Another important component of naturalness is not semantic, though, but simply “how you say it”. This is what has been called collocational knowledge, like the fact that one says *strong tea* and *powerful car*, but not vice versa [23], or that you say *thick accent* in English, but *starker Akzent* (“strong accent”) in German.

Though it is difficult to define just what naturalness is, it is not difficult to recognize it. If one generates text at random from an explicit grammar plus lexicon, the shortcomings of the grammar are immediately obvious in the unnatural—even if not ungrammatical—sentences that are produced. It is also clear that naturalness is not at all the same thing as meaningfulness. For example, I think it is clear that *differential structure* is more natural than *differential child*, even though I could not say what a differential structure might be. Or consider the following examples, that were in fact generated at random from a grammar:

- (5) a. matter-like, complete, alleged strips
a stratigraphic, dubious scattering
a far alternative shallow model
- b. indirect photographic-drill sources
earlier stratigraphically precise minimums
Europe’s cyclic existence

All these examples are about on a par as concerns meaningfulness, but I think the (b) examples are rather more natural than the (a) examples.

Collocations and selectional restrictions have been two important areas of application of statistical methods in computational linguistics. Questions of interest have been both how to include them in a global measure of goodness, and how to induce them distributionally [19], both as a tool for investigations, and as a model of human learning.

Structural preferences. Structural preferences, or parsing strategies, have already been mentioned. A “longest-match” preference is one example. The example

- (6) the emergency crews hate most is domestic violence

is a garden-path because of a strong preference for the longest initial NP, *the emergency crews*, rather than the correct alternative, *the emergency*. (The correct interpretation is: *the emergency [that crews hate most] is domestic violence*.) The longest-match preference plays an important role in the dispreference for the structure (3) that we examined earlier.

As already mentioned, these preferences can be seen as structural preferences, rather than parsing preferences. They interact with the other factors we have been examining in a global measure of goodness. For example, in (6), an even longer match, *the emergency crews hate*, is actually possible, but it violates the dispreference for having plural nouns as nominal modifiers.

Error tolerance. A remarkable property of human language comprehension is its error tolerance. Many sentences that an algebraic grammar would simply classify as ungrammatical are actually perceived to have a particular structure. A simple example is *we sleeps*, a sentence whose intended structure is obvious, albeit ungrammatical. In fact, an erroneous structure may actually be preferred to a grammatical analysis; consider

(7) Thanks for all you help.

which I believe is preferentially interpreted as an erroneous version of *Thanks for all your help*. However, there is a perfectly grammatical analysis: *thanks for all those who you help*.

We can make sense of this phenomenon by supposing that a range of error-correction operations are available, though their application imposes a certain cost. This cost is combined with the other factors we have discussed, to determine a global goodness, and the best analysis is chosen. In (7), the cost of error correction is apparently less than the cost of the alternative in unnaturalness or structural dispreference. Generally, error detection and correction are a major selling point for statistical methods. They were primary motivations for Shannon's noisy channel model [21], which provides the foundation for many computational linguistic techniques.

Learning on the fly. Not only is the language that one is exposed to full of errors, it is produced by others whose grammars and lexica vary from one's own. Frequently, sentences that one encounters can only be analysed by adding new constructions or lexical entries. For example, when the average person hears *a hectare is a hundred ares*, they deduce that *are* is a noun, and succeed in parsing the sentence. But there are limits to learning on the fly, just as there are limits to error correction. Learning on the fly does not help one parse *the a are of I*.

Learning on the fly can be treated much like error correction. The simplest approach is to admit a space of learning operations—e.g., assigning a new part of speech to a word, adding a new subcategorization frame to a verb, etc.—and

assign a cost to applications of the learning operations. In this way it is conceptually straightforward to include learning on the fly in a global optimization.

People are clearly capable of error correction and learning on the fly; they are highly desirable abilities given the noise and variance in the typical linguistic environment. They greatly exacerbate the problem of picking out the intended parse for a sentence, because they explode the candidate space even beyond the already large set of candidates that the grammar provides. To explain how it is nonetheless possible to identify the intended parse, there is no serious alternative to the use of weighted grammars.

Lexical acquisition. A final factor that exacerbates the problem of identifying the correct parse is the sheer richness of natural language grammars and lexica. A goal of earlier linguistic work, and one that is still a central goal of the linguistic work that goes on in computational linguistics, is to develop grammars that assign a reasonable syntactic structure to every sentence of English, or as nearly every sentence as possible. This is not a goal that is currently much in fashion in theoretical linguistics. Especially in GB, the development of large fragments has long since been abandoned in favor of the pursuit of deep principles of grammar.

The scope of the problem of identifying the correct parse cannot be appreciated by examining behavior on small fragments, however deeply analyzed. Large fragments are not just small fragments several times over—there is a qualitative change when one begins studying large fragments. As the range of constructions that the grammar accommodates increases, the number of undesired parses for sentences increases dramatically.

In-breadth studies also give a different perspective on the problem of language acquisition. When one attempts to give a systematic account of phrase structure, it becomes clear just how many little facts there are that do not fall out from grand principles, but just have to be learned. The simple, general principles in these cases are not principles of syntax, but principles of acquisition. Examples are the complex constraints on sequencing of prenominal elements in English, or the syntax of date expressions (*Monday June the 4th*, *Monday June 4*, **Monday June the 4*, **June 4 Monday*) or the syntax of proper names (*Greene County Sheriff's Deputy Jim Thurmond*), or the syntax of numeral expressions.

The largest piece of what must be learned is the lexicon. If parameter-setting views of syntax acquisition are correct, then learning the syntax (which in this case does not include the low-level messy bits discussed in the previous paragraph) is actually almost trivial. The really hard job is learning the lexicon.

Acquisition of the lexicon is a primary area of application for distributional and statistical approaches to acquisition. Methods have been developed for the acquisition of parts of speech [4, 20], terminological noun compounds [1], collocations [23], support verbs [10], subcategorization frames [2, 16], selectional

restrictions [19], and low-level phrase structure rules [7, 24]. These distributional techniques do not so much compete with parameter setting as a model of acquisition, as much as complement it, by addressing issues that parameter-setting accounts pass over in silence. Distributional techniques are also not adequate alone as models of human acquisition—whatever the outcome of the syntactic versus semantic bootstrapping debate, children clearly do make use of situations and meaning to learn language—but the effectiveness of distributional techniques indicates at least that they might account for a component of human language learning.

4 Objections

There are a couple of general objections to statistical methods that may be lurking in the backs of readers minds, that I would like to address. First is the sentiment that, however relevant and effective statistical methods may be, they are no more than an engineer’s approximation, not part of a proper scientific theory. Second is the nagging doubt: didn’t Chomsky debunk all this ages ago?

4.1 Stochastic models are for engineers?

One might admit that one can account for parsing preferences by a probabilistic model, but insist that a probabilistic model is at best an approximation, suitable for engineering but not for science. On this view, we do not need to talk about degrees of grammaticality, or preferences, or degrees of plausibility. Granted, humans perceive only one of the many legal structures for a given sentence, but the perception is completely deterministic. We need only give a proper account of all the factors affecting the judgment.

Consider the example:

Yesterday three shots were fired at Humberto Calvados, personal assistant to the famous tenor Enrique Felicidad, who was in Paris attending to unspecified personal matters.

Suppose for argument’s sake that 60% of readers take the tenor to be in Paris, and 40% take the assistant to be in Paris. Or more to the point, suppose a particular informant, John Smith, chooses the low attachment 60% of the time when encountering sentences with precisely this structure (in the absence of an informative context), and low attachment 40% of the time. One could still insist that no probabilistic decision is being made, but rather that there are lexical and semantic differences that we have inappropriately conflated across sentences with ‘precisely this structure’, and if we take account of these other effects, we end up with a deterministic model after all. A probabilistic model is only a stopgap in absence of an account of the missing factors: semantics, pragmatics, what topics I’ve been talking to other people about lately, how tired I am, whether I ate breakfast this morning.

By this species of argument, stochastic models are practically always a stop-gap approximation. Take stochastic queue theory, for example, by which one can give a probabilistic model of how many trucks will be arriving at given depots in a transportation system. One could argue that if we could just model everything about the state of the trucks and the conditions of the roads, the location of every nail that might cause a flat and every drunk driver that might cause an accident, then we could in principle predict deterministically how many trucks will be arriving at any depot at any time, and there is no need of stochastic queue theory. Stochastic queue theory is only an approximation in lieu of information that it is impractical to collect.

But this argument is flawed. If we have a complex deterministic system, and if we have access to the initial conditions in complete detail, so that we can compute the state of the system unerringly at every point in time, a simpler stochastic description may still be more insightful. To use a dirty word, some properties of the system are genuinely *emergent*, and a stochastic account is not just an approximation, it provides more insight than identifying every deterministic factor. Or to use a different dirty word, it is a *reductionist* error to reject a successful stochastic account and insist that only a more complex, lower-level, deterministic model advances scientific understanding.

4.2 Chomsky v. Shannon

In one's introductory linguistics course, one learns that Chomsky disabused the field once and for all of the notion that there was anything of interest to statistical models of language. But one usually comes away a little fuzzy on the question of what, precisely, he proved.

The arguments of Chomsky's that I know are from "Three Models for the Description of Language" [5] and *Syntactic Structures* [6] (essentially the same argument repeated in both places), and from the *Handbook of Mathematical Psychology*, chapter 13 [17]. I think the first argument in *Syntactic Structures* is the best known. It goes like this.

Neither (a) 'colorless green ideas sleep furiously' nor (b) 'furiously sleep ideas green colorless', nor any of their parts, has ever occurred in the past linguistic experience of an English speaker. But (a) is grammatical, while (b) is not.

This argument only goes through if we assume that if the frequency of a sentence or 'part' is zero in a training sample, its probability is zero. But in fact, there is quite a literature on how to estimate the probabilities of events that do not occur in the sample, and in particular how to distinguish real zeros from zeros that just reflect something that is missing by chance.

Chomsky also gives a more general argument:

If we rank the sequences of a given length in order of statistical approximation to English, we will find both grammatical and ungrammatical sequences scattered throughout the list; there appears to be no particular relation between order of approximation and grammaticalness.

Because for any n , there are sentences with grammatical dependencies spanning more than n words, so that no n th-order statistical approximation can sort out the grammatical from the ungrammatical examples. In a word, you cannot define grammaticality in terms of probability.

It is clear from context that ‘statistical approximation to English’ is a reference to n th-order Markov models, as discussed by Shannon. Chomsky is saying that there is no way to choose n and ϵ such that

$$\text{for all sentences } s, \text{ grammatical}(s) \leftrightarrow P_n(s) > \epsilon$$

where $P_n(s)$ is the probability of s according to the ‘best’ n th-order approximation to English.

But Shannon himself was careful to call attention to precisely this point: that for any n , there will be some dependencies affecting the well-formedness of a sentence that an n th-order model does not capture. The point of Shannon’s approximations is that, as n increases, the total mass of ungrammatical sentences that are erroneously assigned nonzero probability decreases. That is, we *can* in fact define grammaticality in terms of probability, as follows:

$$\text{grammatical}(s) \leftrightarrow \lim_{n \rightarrow \infty} P_n(s) > 0$$

A third variant of the argument appears in the *Handbook*. There Chomsky states that parameter estimation is impractical for an n th-order Markov model where n is large enough “to give a reasonable fit to ordinary usage”. He emphasizes that the problem is not just an inconvenience for statisticians, but renders the model untenable as a model of human language acquisition: “we cannot seriously propose that a child learns the values of 10^9 parameters in a childhood lasting only 10^8 seconds.”

This argument is also only partially valid. If it takes at least a second to estimate each parameter, and parameters are estimated sequentially, the argument is correct. But if parameters are estimated in parallel, say, by a high-dimensional iterative or gradient-pursuit method, all bets are off. Nonetheless, I think even the most hardcore statistical types are willing to admit that Markov models represent a brute force approach, and are not an adequate basis for psychological models of language processing.

However, the inadequacy of Markov models is not that they are statistical, but that they are statistical versions of finite-state automata! Each of Chomsky’s arguments turns on the fact that Markov models are finite-state, not on the fact that they are stochastic. None of his criticisms are applicable

to stochastic models generally. More sophisticated stochastic models do exist: stochastic context-free grammars are well understood, and stochastic versions of Tree-Adjoining Grammar [18], GB [8], and HPSG [3] have been proposed.

In fact, probabilities make Markov models more adequate than their non-probabilistic counterparts, not less adequate. Markov models are surprisingly effective, given their finite-state substrate. For example, they are the workhorse of speech recognition technology. Stochastic grammars can also be easier to learn than their non-stochastic counterparts. For example, though Gold [9] showed that the class of context-free grammars is not learnable, Horning [13] showed that the class of stochastic context-free grammars *is* learnable.

In short, Chomsky's arguments do not bear at all on the probabilistic nature of Markov models, only on the fact that they are finite-state. His arguments are not by any stretch of the imagination a sweeping condemnation of statistical methods.

5 Conclusion

In closing, let me repeat the main line of argument as concisely as I can. Statistical methods—by which I mean primarily weighted grammars and distributional induction methods—are clearly relevant to language acquisition, language change, language variation, language generation, and language comprehension. Understanding language in this broad sense is the ultimate goal of linguistics.

The issues to which weighted grammars apply, particularly as concerns perception of grammaticality and ambiguity, one may be tempted to dismiss as performance issues. However, the set of issues labelled “performance” are not essentially computational, as one is often led to believe. Rather, “competence” represents a provisional narrowing and simplification of data in order to understand the algebraic properties of language. “Performance” is a misleading term for “everything else”. Algebraic methods are inadequate for understanding many important properties of human language, such as the measure of goodness that permits one to identify the correct parse out of a large candidate set in the face of considerable noise.

Many other properties of language, as well, that are mysterious given unweighted grammars, properties such as the gradualness of rule learning, the gradualness of language change, dialect continua, and statistical universals, make a great deal more sense if we assume weighted or stochastic grammars. There is a huge body of mathematical techniques that computational linguists have begun to tap, yielding tremendous progress on previously intractable problems. The focus in computational linguistics has admittedly been on technology. But the same techniques promise progress at long last on questions about the nature of language that have been mysterious for so long. The time is ripe to apply them.

References

- [1] Didier Bourigault. Surface grammatical analysis for the extraction of terminological noun phrases. In *COLING-92, Vol. III*, pages 977–981, 1992.
- [2] Michael R. Brent. Automatic acquisition of subcategorization frames from untagged, free-text corpora. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 209–214, 1991.
- [3] Chris Brew. Stochastic HPSG. In *Proceedings of EACL-95*, 1995.
- [4] Eric Brill. *Transformation-Based Learning*. PhD thesis, Univ. of Pennsylvania, 1993.
- [5] Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, IT-2(3):113–124, 1956. Institute of Radio Engineers, New York.
- [6] Noam Chomsky. *Syntactic Structures*. Mouton, 1957.
- [7] Steven Paul Finch. *Finding Structure in Language*. PhD thesis, University of Edinburgh, 1993.
- [8] Andrew Fordham and Matthew Crocker. Parsing with principles and probabilities. In *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, 1994.
- [9] E. Mark Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.
- [10] Gregory Grefenstette. Corpus-based method for automatic identification of support verbs for nominalizations. In *EACL-95*, 1995.
- [11] John A. Hawkins. *Word Order Universals*. Academic Press, New York, 1983.
- [12] John A. Hawkins. A parsing theory of word order universals. *Linguistic Inquiry*, 21(2):223–262, 1990.
- [13] James Jay Horning. *A Study of Grammatical Inference*. PhD thesis, Stanford (Computer Science), 1969.
- [14] Terrence Kaufman. The native languages of Latin America: general remarks. In Christopher Moseley and R.E. Asher, editors, *Atlas of the World's Languages*, pages 31–33. Routledge, London and New York, 1994.
- [15] Brett Kessler. Computational dialectology in Irish Gaelic. In *EACL-95*, 1995.

- [16] Christopher D. Manning. Automatic acquisition of a large subcategorization dictionary from corpora. In *31st Annual Meeting of the Association for Computational Linguistics*, pages 235–242, 1993.
- [17] George A. Miller and Noam Chomsky. Finitary models of language users. In R.D. Luce, R. Bush, and E. Galanter, editors, *Handbook of Mathematical Psychology*, chapter 13. Wiley, New York, 1963.
- [18] Philip Resnik. Probabilistic Tree-Adjoining Grammar as a framework for statistical natural language processing. In *COLING-92*, pages 418–424, 1992.
- [19] Philip Resnik. *Selection and Information*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1993.
- [20] Hinrich Schütze. Part-of-speech induction from scratch. In *31st Annual Meeting of the Association for Computational Linguistics*, pages 251–258, 1993.
- [21] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3–4):379–423, 623–656, 1948.
- [22] Frank Smadja. Microcoding the lexicon for language generation. In Uri Zernik, editor, *Lexical Acquisition: Using on-line resources to build a lexicon*. MIT Press, 1989.
- [23] Frank Smadja. *Extracting Collocations from Text. An Application: Language Generation*. PhD thesis, Columbia University, New York, NY, 1991.
- [24] Tony C. Smith and Ian H. Witten. Language inference from function words. Manuscript, University of Calgary and University of Waikato, January 1993.
- [25] Whitney Tabor. The gradualness of syntactic change: A corpus proximity model. Ms. for Berkeley Colloquium talk, CSLI, Stanford University, November 1993.
- [26] Mary Tait. North America. In Christopher Moseley and R.E. Asher, editors, *Atlas of the World's Languages*, pages 3–30. Routledge, London and New York, 1994.
- [27] J.R.R. Tolkien. *The Hobbit*. Houghton Mifflin Co., Boston, 1966.
- [28] Willard van Orman Quine. *Word and Object*. The MIT Press, Cambridge, MA, 1960.

Stochastic Attribute-Value Grammars

Steven P. Abney*
AT&T Laboratories

Probabilistic analogues of regular and context-free grammars are well-known in computational linguistics, and currently the subject of intensive research. To date, however, no satisfactory probabilistic analogue of attribute-value grammars has been proposed: previous attempts have failed to define an adequate parameter-estimation algorithm.

In the present paper, I define stochastic attribute-value grammars and give an algorithm for computing the maximum-likelihood estimate of their parameters. The estimation algorithm is adapted from (Della Pietra, Della Pietra, and Lafferty, 1995). To estimate model parameters, it is necessary to compute the expectations of certain functions under random fields. In the application discussed by Della Pietra, Della Pietra, and Lafferty (representing English orthographic constraints), Gibbs sampling can be used to estimate the needed expectations. The fact that attribute-value grammars generate constrained languages makes Gibbs sampling inapplicable, but I show that sampling can be done using the more general Metropolis-Hastings algorithm.

1. Introduction

Stochastic versions of regular grammars and context-free grammars have received a great deal of attention in computational linguistics for the last several years, and basic techniques of stochastic parsing and parameter estimation have been known for decades. However, regular and context-free grammars are widely deemed linguistically inadequate; standard grammars in computational linguistics are attribute-value (AV) grammars of some variety. Before the advent of statistical methods, regular and context-free grammars were considered too inexpressive for serious consideration, and even now the reliance on stochastic versions of the less-expressive grammars is often seen as an expedient necessitated by the lack of an adequate stochastic version of attribute-value grammars.

Proposals have been made for extending stochastic models developed for the regular

* AT&T Laboratories, Rm. A216, 180 Park Avenue, Florham Park, NJ 07932

and context-free cases to grammars with constraints.¹ (Brew, 1995) sketches a probabilistic version of Head-Driven Phrase Structure Grammar (HPSG). He proposes a stochastic process for generating attribute-value structures, that is, directed acyclic graphs (dags). A dag is generated starting from a single node labelled with the (unique) most general type. Each type S has a set of *maximal subtypes* T_1, \dots, T_m . To expand a node labelled S , one chooses a maximal subtype T stochastically. One then considers equating the current node with other nodes of type T , making a stochastic yes/no decision for each. Equating two nodes creates a re-entrancy. If the current node is equated with no other node, one proceeds to expand it. Each maximal type *introduces* types U_1, \dots, U_n , corresponding to values of attributes; one creates a child node for each introduced type, and then expands each child in turn. A limitation of this approach is that it permits one to specify only the average rate of re-entrancies; it does not permit one to specify more complex context dependencies.

(Eisele, 1994) takes a logic-programming approach to constraint grammars. He assigns probabilities to proof trees by attaching parameters to logic program clauses. He presents the following logic program as an example:

1. $p(X, Y, Z) \leftarrow_1 q(X, Y), r(Y, Z).$
2. $q(a, b) \leftarrow_{0.4} .$
3. $q(X, c) \leftarrow_{0.6} .$
4. $r(b, d) \leftarrow_{0.5} .$
5. $r(X, e) \leftarrow_{0.5} .$

The probability of a proof tree is defined to be proportional to the product of the probabilities of clauses used in the proof. Normalization is necessary, because some derivations lead to invalid proof trees: for example, the derivation

¹ I confine my discussion here to Brew and Eisele because they aim to describe parametric models of probability distributions over the languages of constraint-based grammars, and to estimate the parameters of those models. Other authors have assigned weights or preferences to constraint-based grammars but not discussed parameter estimation. One approach of the latter sort that I find of particular interest is that of Stefan Riezler (Riezler, 1996), who describes a weighted logic for constraint-based grammars that characterizes the languages of the grammars as fuzzy sets. This interpretation avoids the need for normalization that Brew and Eisele face, though parameter estimation still remains to be addressed.

$$p(X, Y, Z) \stackrel{\text{by } 1}{\leftarrow} q(X, Y) \quad r(Y, Z) \stackrel{\text{by } 3}{\leftarrow} r(c, Z) : Y=c \stackrel{\text{by } 4}{\leftarrow} : Y=c \quad b=c \quad Z=d$$

is invalid because of the illegal assignment $b = c$.

Both Brew and Eisele associate weights with analogues of rewrite rules. In Brew's case, we can view type expansion as a stochastic choice from a finite set of rules of form $X \rightarrow \xi_i$, where X is the type to expand and each ξ_i is a sequence of introduced child types. A re-entrancy decision is a stochastic choice between two rules, $X \rightarrow \text{yes}$ and $X \rightarrow \text{no}$, where X is the type of the node being considered for re-entrancy. In Eisele's case, expanding a goal term can be viewed as a stochastic choice among a finite set of rules $X \rightarrow \xi_i$, where X is the predicate of the goal term and each ξ_i is a program clause whose head has predicate X . The parameters of the models are essentially weights on such rules, representing the probability of choosing ξ_i when making a choice of type X .

In these terms, Brew and Eisele propose estimating parameters as the empirical relative frequency of the corresponding rules. That is, the weight of the rule $X \rightarrow \xi_i$ is obtained by counting the number of times X rewrites as ξ_i in the training corpus, divided by the total number of times X is rewritten in the training corpus. For want of a standard term, let us call these estimates Empirical Relative Frequency (ERF) estimates. To deal with incomplete data, both Brew and Eisele appeal to the Expectation-Maximization (EM) algorithm, applied however to ERF rather than maximum likelihood estimates.

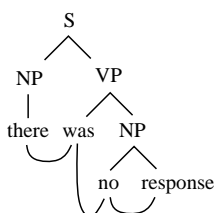
Under certain independence conditions, ERF estimates *are* maximum likelihood estimates. Unfortunately, these conditions are violated when there are context dependencies of the sort found in attribute-value grammars, as will be shown below. As a consequence, applying the ERF method to attribute-value grammars does *not* generally yield maximum likelihood estimates. This is true whether one uses EM or not—a method that yields the “wrong” estimates on complete data does not improve when EM is used to extend the method to incomplete data.

Eisele identifies an important symptom that something is amiss with ERF estimates: the probability distribution over proof trees that one obtains does not agree with the frequency of proof trees in the training corpus. Eisele recognizes that this problem arises only where there are context dependencies.

Fortunately, solutions to the context-dependency problem have been described (and indeed are currently enjoying a surge of interest) in statistics, machine learning, and statistical pattern recognition, particularly image processing. The models of interest are known as random fields. Random fields can be seen as a generalization of Markov chains and stochastic branching processes. Markov chains are stochastic processes corresponding to regular grammars and random branching processes are stochastic processes corresponding to context-free grammars. The evolution of a Markov chain describes a line, in which each stochastic choice depends only on the state at the immediately preceding time-point. The evolution of a random branching process describes a tree in which a finite-state process may spawn multiple child processes at the next time-step, but the number of processes and their states depend only on the state of the unique parent process at the preceding time-step. In particular, stochastic choices are *independent* of other choices at the same time-step: each process evolves independently. If we permit re-entrancies, that is, if we permit processes to re-merge, we generally introduce context-sensitivity. In order to re-merge, processes must be “in synch,” which is to say, they cannot evolve in complete independence of one another. Random fields are a particular class of multi-dimensional random processes, that is, processes corresponding to probability distributions over an arbitrary graph. The theory of random fields can be traced back to (Gibbs, 1902); indeed, the probability distributions involved are known as Gibbs distributions.

To my knowledge, the first application of random fields to natural language was (Mark et al., 1992). The problem of interest was how to combine a stochastic context-free grammar with n -gram language models. In the resulting structures, the probability

of choosing a particular word is constrained simultaneously by the syntactic tree in which it appears and the choices of words at the n preceding positions. The context-sensitive constraints introduced by the n -gram model are reflected in re-entrancies in the structure of statistical dependencies, e.g.:



In this diagram, the choice of label on a node z with parent x and preceding word y is dependent on the label of x and y , but conditionally independent of the label on any other node.

(Della Pietra, Della Pietra, and Lafferty, 1995, henceforth, DD&L) also apply random fields to natural language processing. The application they consider is the induction of English orthographic constraints—inducing a grammar of possible English words. DD&L describe an algorithm called Improved Iterative Scaling (IIS) for selecting informative features of words to construct a random field, and for setting the parameters of the field optimally for a given set of features, to model an empirical word distribution.

It is not immediately obvious how to use the IIS algorithm to equip attribute-value grammars with probabilities. In brief, the difficulty is the following. The IIS algorithm requires the computation of the expectations, under random fields, of certain functions. In general, computing these expectations involves summing over all configurations (all possible character sequences, in the orthography application), which is not possible when the configuration space is large. Instead, DD&L use Gibbs sampling to estimate the needed expectations.

Gibbs sampling is possible for the application that DD&L consider. A prerequisite for Gibbs sampling is that the configuration space be closed under relabelling of graph

nodes. In the orthography application, the configuration space is the set of possible English words, represented as finite linear graphs labelled with ASCII characters. Every way of changing a label, that is, every substitution of one ASCII character for a different one, yields a possible English word.

By contrast, the set of graphs admitted by an attribute-value grammar G is highly constrained. If one changes an arbitrary node label in a dag admitted by G , one does *not* necessarily obtain a new dag admitted by G . Hence, Gibbs sampling is not applicable. However, I will show that a more general sampling method, the Metropolis-Hastings algorithm, can be used to compute the maximum-likelihood estimate of the parameters of AV grammars.

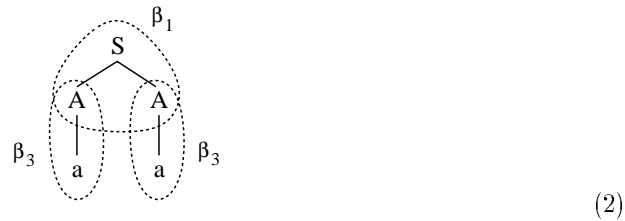
2. Stochastic Context-Free Grammars

Let us begin by examining stochastic context-free grammars (SCFGs) and asking why the natural extension of SCFG parameter estimation to attribute-value grammars fails. A point of terminology: I will use the term *grammar* to refer to an unweighted grammar, be it a context-free grammar or attribute-value grammar. A grammar equipped with weights (and other periphenalia as necessary) I will refer to as a *model*. Occasionally I will also use *model* to refer to the weights themselves, or the probability distribution they define.

Throughout we will use the following stochastic context-free grammar for illustrative purposes. Let us call the underlying grammar G_1 and the grammar equipped with weights as shown, M_1 :

1. $S \rightarrow A A$ $\beta_1 = 1/2$
 2. $S \rightarrow B$ $\beta_2 = 1/2$
 3. $A \rightarrow a$ $\beta_3 = 2/3$
 4. $A \rightarrow b$ $\beta_4 = 1/3$
 5. $B \rightarrow a a$ $\beta_5 = 1/2$
 6. $B \rightarrow b b$ $\beta_6 = 1/2$
- (1)

The probability of a given tree is computed as the product of probabilities of rules used in it. For example:



Let x be tree (2) and let q_1 be the probability distribution over trees defined by model M_1 . Then:

$$q_1(x) = \beta_1 \cdot \beta_3 \cdot \beta_3 = \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{2}{3} = \frac{2}{9}$$

In parsing, we use the probability distribution $q_1(x)$ defined by model M_1 to disambiguate: the grammar assigns some set of trees $\{x_1, \dots, x_n\}$ to a sentence σ , and we choose that tree x_i that has greatest probability $q_1(x_i)$. The issue of efficiently computing the most-probable parse for a given sentence has been thoroughly addressed in the literature. The standard parsing techniques can be readily adapted to the random-field models to be discussed below, so I simply refer the reader to the literature. Instead, I concentrate on parameter estimation, which for attribute-value grammars cannot be accomplished by standard techniques.

By *parameter estimation* we mean determining values for the weights β . In order for a stochastic grammar to be useful, we must be able to compute the correct weights, where by *correct weights* we mean the weights that best account for a training corpus.

The degree to which a given set of weights accounts for a training corpus is measured by the similarity between the distribution $q(x)$ determined by the weights β and the distribution of trees x in the training corpus.

2.1 The Goodness of a Model

The distribution determined by the training corpus is known as the **empirical distribution**. For example, suppose we have a training corpus containing twelve trees of the following four types from $L(G_1)$:

x_1	x_2	x_3	x_4	
$\begin{array}{c} \text{S} \\ / \quad \backslash \\ \text{A} \quad \text{A} \\ \quad \\ \text{a} \quad \text{a} \end{array}$	$\begin{array}{c} \text{S} \\ / \quad \backslash \\ \text{A} \quad \text{A} \\ \quad \\ \text{b} \quad \text{b} \end{array}$	$\begin{array}{c} \text{S} \\ \\ \text{B} \\ / \quad \backslash \\ \text{a} \quad \text{a} \end{array}$	$\begin{array}{c} \text{S} \\ \\ \text{B} \\ / \quad \backslash \\ \text{b} \quad \text{b} \end{array}$	
$c = 4x$	$2x$	$3x$	$3x$	$= 12$
$\tilde{p} = 4/12$	$2/12$	$3/12$	$3/12$	(3)

where $c(x)$ is the count of how often the tree (type) x appears in the corpus, and $\tilde{p}(\cdot)$ is the empirical distribution, defined as:

$$\tilde{p}(x) = \frac{c(x)}{N} \quad N = \sum_x c(x)$$

In comparing a distribution q to the empirical distribution \tilde{p} , we shall actually measure dissimilarity rather than similarity. Our measure for dissimilarity of distributions is the Kullback-Leibler (KL) divergence, defined as:

$$D(\tilde{p}||q) = \sum_x \tilde{p}(x) \ln \frac{\tilde{p}(x)}{q(x)}$$

The divergence between \tilde{p} and q at point x is the log of the ratio of $\tilde{p}(x)$ to $q(x)$. The overall divergence between \tilde{p} and q is the average divergence, where the averaging is over tree (tokens) in the corpus; i.e., point divergences $\ln \tilde{p}(x)/q(x)$ are weighted by $\tilde{p}(x)$ and summed.

For example, let q_1 be, as before, the distribution determined by model M_1 . The following table shows q_1 , \tilde{p} , the ratio $q_1(x)/\tilde{p}(x)$, and the weighted point divergence $\tilde{p}(x) \ln(\tilde{p}(x)/q_1(x))$. The sum of the fourth column is the KL divergence $D(\tilde{p}\|q_1)$ between \tilde{p} and q_1 . The third column contains $q_1(x)/\tilde{p}(x)$ rather than $\tilde{p}(x)/q_1(x)$ so that one can see at a glance whether $q_1(x)$ is too large (> 1) or too small (< 1).

	q_1	\tilde{p}	q_1/\tilde{p}	$\tilde{p} \ln(\tilde{p}/q_1)$	
x_1	2/9	1/3	0.67	0.14	
x_2	1/18	1/6	0.33	0.18	
x_3	1/4	1/4	1.00	0.00	(4)
x_4	1/4	1/4	1.00	0.00	
				0.32	

The total divergence $D(\tilde{p}\|q_1) = 0.32$.

One set of weights is better than another if its divergence from the empirical distribution is less. For example, let us consider a different set of weights for grammar G_1 . Let M' be G_1 with weights $(1/2, 1/2, 1/2, 1/2, 1/2, 1/2)$, and let q' be the probability distribution determined by M' . Then the computation of the KL divergence is as follows:

	q'	\tilde{p}	q'/\tilde{p}	$\tilde{p} \ln(\tilde{p}/q')$
x_1	1/8	1/3	0.38	0.33
x_2	1/8	1/6	0.75	0.05
x_3	1/4	1/4	1.00	0.00
x_4	1/4	1/4	1.00	0.00
				0.38

The fit for x_2 improves, but that is more than offset by a poorer fit for x_1 . The distribution q_1 is a better distribution than q' , in the sense that q_1 is more similar (less dissimilar) to the empirical distribution than q' is.

One reason for adopting minimal KL divergence as a measure of goodness is that minimizing KL divergence maximizes likelihood. The likelihood of distribution q is the probability of the training corpus according to q :

$$\begin{aligned} L(q) &= \prod_{x \text{ in training}} q(x) \\ &= \prod_x q(x)^{c(x)} \end{aligned}$$

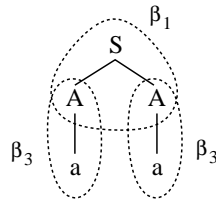
Since log is monotone increasing, maximizing likelihood is equivalent to maximizing log likelihood:

$$\begin{aligned}\ln L(q) &= \sum_x c(x) \ln q(x) \\ &= N \sum_x \tilde{p}(x) \ln q(x)\end{aligned}$$

The expression on the right hand side is $-1/N$ times the cross entropy of q with respect to \tilde{p} , hence maximizing log likelihood is equivalent to minimizing cross entropy. Finally, $D(\tilde{p}\|q)$ is equal to the cross entropy of q less the entropy of \tilde{p} , and the entropy of \tilde{p} is constant with respect to q ; hence minimizing cross entropy (maximizing likelihood) is equivalent to minimizing divergence.

2.2 The ERF Method

For stochastic context-free grammars, it can be shown that the ERF method yields the best model for a given training corpus. First, let us introduce some terminology and notation. With each rule i in a stochastic context-free grammar is associated a weight β_i and a function $f_i(x)$ that returns the number of times rule i is used in the derivation of tree x . For example, consider tree (2), repeated here for convenience:



Rule 1 is used once and rule 3 is used twice; accordingly $f_1(x) = 1$, $f_3(x) = 2$, and $f_i(x) = 0$ for $i \in \{2, 4, 5, 6\}$.

We use the notation $p[f]$ to represent the expectation of f under probability distribution p ; that is, $p[f] = \sum_x p(x)f(x)$. The ERF method instructs us to choose the weight β_i for rule i proportional to its empirical expectation $\tilde{p}[f_i]$. Algorithmically, we compute the expectation of each rule's frequency, and normalize among rules with the same lefthand side.

To illustrate, let us consider corpus (3) again. The expectation of each rule frequency f_i is a sum of terms $\tilde{p}(x) f_i(x)$. These terms are shown for each tree, in the following table.

		\tilde{p}	$S \rightarrow A A$ $\tilde{p}f_1$	$S \rightarrow B$ $\tilde{p}f_2$	$A \rightarrow a$ $\tilde{p}f_3$	$A \rightarrow b$ $\tilde{p}f_4$	$B \rightarrow a a$ $\tilde{p}f_5$	$B \rightarrow b b$ $\tilde{p}f_6$
x_1	$[S [A a] [A a]]$	1/3	1/3		2/3			
x_2	$[S [A b] [A b]]$	1/6	1/6			2/6		
x_3	$[S [B a a]]$	1/4		1/4			1/4	
x_4	$[S [B b b]]$	1/4		1/4				1/4
	$\tilde{p}[f] =$		1/2	1/2	2/3	1/3	1/4	1/4
	$\beta =$		1/2	1/2	2/3	1/3	1/2	1/2

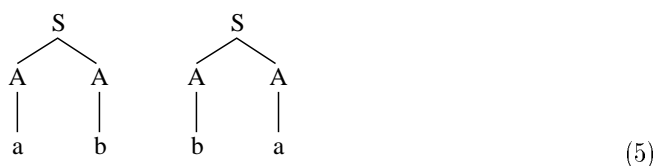
For example, in tree x_1 , rule 1 is used once and rule 3 is used twice. The empirical probability of x_1 is 1/3, so x_1 's contribution to $\tilde{p}[f_1]$ is 1/3 · 1, and its contribution to $\tilde{p}[f_3]$ is 1/3 · 2. The weight β_i is obtained from $\tilde{p}[f_i]$ by normalizing among rules with the same lefthand side. For example, the expected rule frequencies $\tilde{p}[f_1]$ and $\tilde{p}[f_2]$ of rules with lefthand side S already sum to 1, so they are adopted without change as β_1 and β_2 . On the other hand, the expected rule frequencies $\tilde{p}[f_5]$ and $\tilde{p}[f_6]$ for rules with lefthand side B sum to 1/2, not 1, so they are doubled to yield weights β_5 and β_6 . It should be observed that the resulting weights are precisely the weights of model M_1 .

It can be proven that the ERF weights are the best weights for a given context-free grammar, in the sense that they define the distribution that is most similar to the empirical distribution. That is, if β are the ERF weights (for a given grammar), defining distribution q , and β' defining q' is any set of weights such that $q \neq q'$, then $D(\tilde{p}||q) < D(\tilde{p}||q')$.

One might expect the best weights to yield $D(\tilde{p}||q) = 0$, but such is not the case. We have just seen, for example, that the best weights for grammar G_1 yield distribution q_1 , yet $D(\tilde{p}||q_1) = 0.32 > 0$. A closer inspection of the divergence calculation (4) reveals that q_1 is sometimes less than \tilde{p} , but never greater than \tilde{p} . Could we improve the fit by increasing q_1 ? For that matter, how can it be that q_1 is never greater than \tilde{p} ? As

probability distributions, q_1 and \tilde{p} should have the same total mass, namely, one. Where is the missing mass for q_1 ?

The answer is of course that q_1 and \tilde{p} are probability distributions over $L(G)$, but not all of $L(G)$ appears in the corpus. Two trees are missing, and they account for the missing mass. These two trees are:



Each of these trees has probability 0 according to \tilde{p} (hence they can be ignored in the divergence calculation), but probability $1/9$ according to q_1 .

Intuitively, the problem is this. The distribution q_1 assigns too little weight to trees x_1 and x_2 , and too much weight to the “missing” trees (5); call them x_5 and x_6 . Yet exactly the same rules are used in x_5 and x_6 as are used in x_1 and x_2 . Hence there is no way to increase the weight for trees x_1 and x_2 , improving their fit to \tilde{p} , without simultaneously increasing the weight for x_5 and x_6 , making their fit to \tilde{p} worse. The distribution q_1 is the best compromise possible.

To say it another way, our assumption that the corpus was generated by a context-free grammar means that any context dependencies in the corpus must be accidental, the result of sampling noise. There is indeed a dependency in corpus (3): in the trees where there are two A’s, the A’s always rewrite the same way. If corpus (3) was generated by a stochastic context-free grammar, then this dependency is accidental.

This does not mean that the context-free assumption is wrong. If we generate twelve trees at random from q_1 , it would not be too surprising if we got corpus (3). More extremely, if we generate a random corpus of size 1 from q_1 , it is quite impossible for the resulting empirical distribution to match the distribution q_1 . But as the corpus size

increases, the fit between \tilde{p} and q_1 becomes ever better.

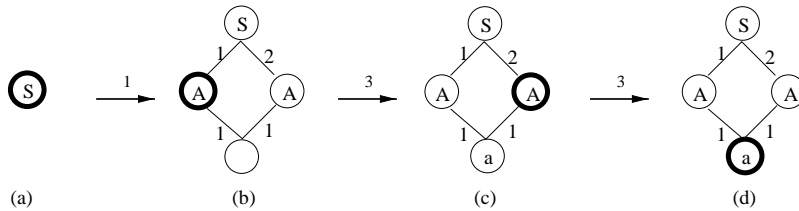
3. Attribute-Value Grammars

But what if the dependency in corpus (3) is not accidental? What if we wish to adopt a grammar that imposes the constraint that both A's rewrite the same way? We can impose such a constraint by means of an attribute-value grammar.

We may formalize an attribute-value grammar as a context-free grammar with attribute labels and path equations. An example is the following grammar; let us call it G_2 :

1. $S \rightarrow 1:A \ 2:A \quad \langle 1 \ 1 \rangle = \langle 2 \ 1 \rangle$
 2. $S \rightarrow 1:B$
 3. $A \rightarrow 1:a$
 4. $A \rightarrow 1:b$
 5. $B \rightarrow 1:a$
 6. $B \rightarrow 1:b$
- (G_2)

The following illustrates how a dag is generated from G_2 .



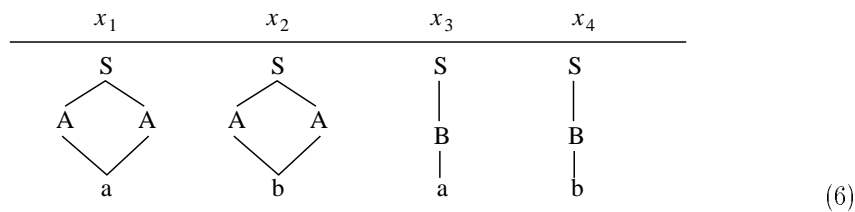
We begin in (a) with a single node labelled with the start category of G_2 , namely, S . A node x is expanded by choosing a rule that rewrites the category of x . In this case, we choose rule 1 to expand the root node. Rule 1 instructs us to create two children, both labelled A . The edge to the first child is labelled "1" and the edge to the second child is labelled "2". The constraint " $\langle 1 \ 1 \rangle = \langle 2 \ 1 \rangle$ " indicates that the "1" child of the "1" child of x is identical to the "1" child of the "2" child of x . We create an unlabelled node to represent this grandchild of x and direct appropriately labelled edges from the children, yielding (b).

We proceed to expand the newly introduced nodes. We choose rule 3 to expand the

first “A” node. In this case, a child with edge labelled “1” already exists, so we use it rather than creating a new one. Rule 3 instructs us to label this child “a”, yielding (c). Now we expand the second “A” node. Again we choose rule 3. We are instructed to label the “1” child “a”, but it already has that label, so we do not need to do anything. Finally, in (d), the only remaining node is the bottommost node, labelled “a”. Since its label is a terminal category, it does not need to be expanded, and we are done.

Let us back up to (c) again. Here we were free to choose rule 4 instead of rule 3 to expand the righthand “A” node. Rule 4 instructs us to label the “1” child “b”, but we cannot, inasmuch as it is already labelled “a”. The derivation fails, and no dag is generated.

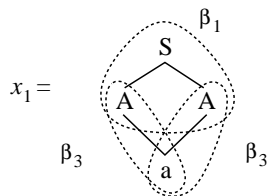
The language $L(G_2)$ is the set of dags produced by successful derivations, namely:



(The edges of the dags should actually be labelled with 1’s and 2’s, but I have suppressed the edge labels for the sake of perspicuity.)

3.1 AV Grammars and the ERF Method

Now we face the question of how to attach probabilities to grammar G_2 . The natural extension of the method we used for context-free grammars is the following. Associate a weight with each of the six rules of grammar G_2 . For example, let M_2 be the model consisting of G_2 plus weights $(\beta_1, \dots, \beta_6) = (1/2, 1/2, 2/3, 1/3, 1/2, 1/2)$. Let $\phi_2(x)$ be the weight that M_2 assigns to dag x ; it is defined to be the product of the weights of the rules used to generate x . For example, the weight $\phi_2(x_1)$ assigned to tree x_1 of (6) is $2/9$, computed as follows:



Rule 1 is used once and rule 3 is used twice; hence $\phi_2(x_1) = \beta_1\beta_3\beta_3 = 1/2 \cdot 2/3 \cdot 2/3 = 2/9$.

Observe that $\phi_2(x_1) = \beta_1\beta_3^2$, which is to say, $\beta_1^{f_1(x_1)}\beta_3^{f_3(x_1)}$. Moreover, since $\beta^0 = 1$, it does not hurt to include additional factors $\beta_i^{f_i(x_1)}$ for those i where $f_i(x_1) = 0$. That is, we can define the dag weight ϕ corresponding to rule weights $\beta = (\beta_1, \dots, \beta_n)$ generally as:

$$\phi(x) = \prod_{i=1}^n \beta_i^{f_i(x)}$$

The next question is how to estimate weights. Let us consider what happens when we use the ERF method. Let us assume a corpus distribution for the dags (6) analogous to the distribution in (3):

$$\tilde{p} = \begin{array}{cccc} & x_1 & x_2 & x_3 & x_4 \\ & 1/3 & 1/6 & 1/4 & 1/4 \end{array} \quad (7)$$

Using the ERF method, we estimate rule weights as follows:

	\tilde{p}	$\tilde{p}f_1$	$\tilde{p}f_2$	$\tilde{p}f_3$	$\tilde{p}f_4$	$\tilde{p}f_5$	$\tilde{p}f_6$
x_1	1/3	1/3		2/3			
x_2	1/6	1/6		2/6			
x_3	1/4		1/4			1/4	
x_4	1/4		1/4				1/4
$\tilde{p}[f] =$		1/2	1/2	2/3	1/3	1/4	1/4
$\beta =$		1/2	1/2	2/3	1/3	1/2	1/2

(8)

This table is identical to the one given earlier in the context-free case. We arrive at the same weights M_2 we considered above, defining dag weights $\phi_2(x)$.

3.2 Why the ERF Method Fails

But at this point a problem arises: ϕ_2 is not a probability distribution. Unlike in the context-free case, the four dags in (6) constitute the entirety of $L(G)$. This time, there

are no missing dags to account for the missing probability mass.

There is an obvious “fix” for this problem: we can simply normalize ϕ_2 . We might define the distribution q for an AV grammar with weight function ϕ as:

$$q(x) = \frac{1}{Z} \phi(x)$$

where Z is the normalizing constant:

$$Z = \sum_{x \in L(G)} \phi(x)$$

In particular, for ϕ_2 , we have $Z = 2/9 + 1/18 + 1/4 + 1/4 = 7/9$. Dividing ϕ_2 by $7/9$ yields the ERF distribution:

$$q_2(x) = \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ 2/7 & 1/14 & 9/28 & 9/28 \end{array}$$

On the face of it, then, we can transplant the methods we used in the context-free case to the AV case and nothing goes wrong. The only problem that arises (ϕ not summing to one) has an obvious fix (normalization).

However, something has actually gone very wrong. The ERF method yields the best weights only under certain conditions that we inadvertently violated by changing $L(G)$ and re-apportioning probability via normalization. In point of fact, we can easily see that the ERF weights (8) are *not* the best weights for our example grammar. Consider the alternative model M^* given in (9), defining probability distribution q^* :

$$\begin{array}{cccccc} S \rightarrow A A & S \rightarrow B & A \rightarrow a & A \rightarrow b & B \rightarrow a & B \rightarrow b \\ \frac{3+2\sqrt{2}}{6+2\sqrt{2}} & \frac{3}{6+2\sqrt{2}} & \frac{\sqrt{2}}{1+\sqrt{2}} & \frac{1}{1+\sqrt{2}} & \frac{1}{2} & \frac{1}{2} \end{array} \quad (9)$$

These weights are proper, in the sense that weights for rules with the same lefthand side sum to one. The reader can verify that ϕ^* sums to $Z = \frac{3+\sqrt{2}}{3}$ and that q^* is:

$$q^*(x) = \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ 1/3 & 1/6 & 1/4 & 1/4 \end{array}$$

That is, $q^* = \tilde{p}$. Comparing q_2 (the ERF distribution) and q^* to \tilde{p} , we observe that $D(\tilde{p}\|q_2) = 0.07$ but $D(\tilde{p}\|q^*) = 0$.

In short, in the AV case, the ERF weights do *not* yield the best weights. This means that the ERF method does *not* converge to the correct weights as the corpus size increases. If there are genuine dependencies in the grammar, the ERF method converges systematically to the wrong weights. Fortunately, there are methods that do converge to the right weights. These are methods that have been developed for random fields.

4. Random Fields

A random field defines a probability distribution over a set of labelled graphs Ω called *configurations*. In our case, the configurations are the dags generated by the grammar, i.e., $\Omega = L(G)$. The weight assigned to a configuration is the product of the weights assigned to selected **features** of the configuration. We use the notation:

$$\phi(x) = \prod_i \beta_i^{f_i(x)}$$

where β_i is the weight for feature i and $f_i(\cdot)$ is its frequency function, that is, $f_i(x)$ is the number of times that feature i occurs in configuration x . (For most purposes, a feature can be identified with its frequency function; I will not always make a careful distinction between them.)

I use the term *feature* here as it is used in the machine learning and statistical pattern recognition literature, *not* as in the constraint grammar literature, where *feature* is synonymous with *attribute*. In my usage, dag edges are labelled with attributes, not features. Features are rather like geographic features of dags: a feature is some larger or smaller piece of structure that occurs—possibly at more than one place—in a dag.

The probability of a configuration (that is, a dag) is proportional to its weight, and is obtained by normalizing the weight distribution.

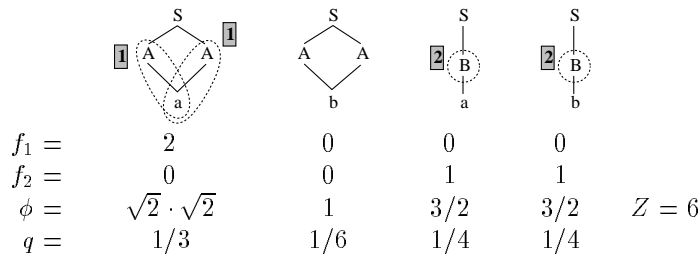
$$q(x) = \frac{1}{Z}\phi(x) \quad Z = \sum_{x \in \Omega} \phi(x)$$

If we identify the features of a configuration with local trees—equivalently, with applications of rewrite rules—the random field model is almost identical to the model we considered in the previous section. There are two important differences. First, we no longer require weights to sum to one for rules with the same lefthand side. Second, the model does not *require* features to be identified with rewrite rules. We use the grammar to define the set of configurations $\Omega = L(G)$, but in defining a probability distribution over $L(G)$, we can choose features of dags however we wish.

Let us consider an example. Let us continue to assume grammar G_2 generating language (6), and let us continue to assume the empirical distribution (7). But now rather than taking rule applications to be features, let us adopt the following two features:



For purpose of illustration, take feature 1 to have weight $\beta_1 = \sqrt{2}$ and feature 2 to have weight $\beta_2 = 3/2$. The functions f_1 and f_2 represent the frequencies of features 1 and 2, respectively:



We are able to exactly recreate the empirical distribution using fewer features than before. Intuitively, we need only use as many features as are necessary to distinguish among trees that have different empirical probabilities.

This added flexibility is welcome, but it does make parameter estimation more involved. Now we must not only choose values for weights, we must also choose the features that weights are to be associated with. We would like to do both in a way that permits us to find the best model, in the sense of the model that minimizes the Kullback-Leibler distance with respect to the empirical distribution. The IIS algorithm (Della Pietra, Della Pietra, and Lafferty, 1995) provides a method to do precisely that.

5. Field Induction

In outline, the IIS algorithm is as follows:

1. Start ($t = 0$) with the null field, containing no features.
2. **Feature Selection.** Consider every feature that might be added to field M_t and choose the best one.
3. **Weight Adjustment.** Readjust weights for all features. The result is field M_{t+1} .
4. Iterate until the field cannot be improved.

For the sake of concreteness, let us take features to be labelled subdags. In step 2 of the algorithm we do not consider every conceivable labelled subdag, but only the atomic (i.e., single-node) subdags and those complex subdags that can be constructed by combining features already in the field or by combining a feature in the field with some atomic feature. We also limit our attention to features that actually occur in the training corpus.

In our running example, the atomic features are:

(S) (A) (B) (a) (b)

Features can be combined by adding connecting arcs. For example:

distribution, we associate a weight with each rule, the weights for rules with a common lefthand side summing to one. The probability of a dag is proportional to the product of weights of rules used to generate it. (Renormalization is necessary because of the failed derivations.) We estimate weights using the ERF method: we estimate the weight of a rule as the relative frequency of the rule in the training corpus, among rules with the same lefthand side.

The resulting initial distribution (the ERF distribution) is not the maximum likelihood distribution, as we know. But it can be taken as a useful first approximation. Intuitively, we begin with the ERF distribution and construct a random field to take account of context-dependencies that the ERF distribution fails to capture, incrementally improving the fit to the empirical distribution.





In this framework, a model consists of: (1) An AV grammar G whose purpose is to define a set of dags $L(G)$. (2) A set of initial weights θ attached to the rules of G . The weight of a dag is the product of weights of rules used in generating it. Discarding failed derivations and renormalizing yields the initial distribution $p_0(x)$. (3) A set of features f_1, \dots, f_n with weights β_1, \dots, β_n to define the field distribution $q(x) = \frac{1}{Z} p_0(x) \prod_i \beta_i^{f_i(x)}$.

5.2 Feature Selection

At each iteration, we select a new feature f by considering all atomic features, and all complex features that can be constructed from features already in the field. Holding the weights constant for all old features in the field, we choose the best weight β for f (how β is chosen will be discussed shortly), yielding a new distribution $q_{\beta,f}$. The score for feature f is the reduction it permits in $D(\tilde{p}\|q_{\text{old}})$, where q_{old} is the old field. That is, the score for f is $D(\tilde{p}\|q_{\text{old}}) - D(\tilde{p}\|q_{\beta,f})$. We compute the score for each candidate feature and add to the field that feature with the highest score.

To illustrate, consider the two atomic features ‘a’ and ‘B’. Given the null field as old

field, the best weight for ‘a’ is $\beta = 7/5$, and the best weight for ‘B’ is $\beta = 1$. This yields q and $D(\tilde{p}\|f)$ as follows:

					
\tilde{p}	1/3	1/6	1/4	1/4	
ϕ_a	7/5	1	7/5	1	$Z = 24/5$
q_a	7/24	5/24	7/24	5/24	
$\tilde{p} \ln \frac{\tilde{p}}{q_a}$	0.04	-0.04	-0.04	0.05	$D = 0.01$
ϕ_B	1	1	1	1	$Z = 4$
q_B	1/4	1/4	1/4	1/4	
$\tilde{p} \ln \frac{\tilde{p}}{q_B}$	0.10	-0.07	0	0	$D = 0.03$

The better feature is ‘a’, and ‘a’ would be added to the field if these were the only two choices.

Intuitively, ‘a’ is better than ‘B’ because ‘a’ permits us to distinguish the set $\{x_1, x_3\}$ from the set $\{x_2, x_4\}$; the empirical probability of the former is $1/3 + 1/4 = 7/12$ whereas the empirical probability of the latter is $5/12$. Distinguishing these sets permits us to model the empirical distribution better (since the old field assigns them equal probability, counter to the empirical distribution). By contrast, the feature ‘B’ distinguishes the set $\{x_1, x_2\}$ from $\{x_3, x_4\}$. The empirical probability of the former is $1/3 + 1/6 = 1/2$ and the empirical probability of the latter is also $1/2$. The old field models these probabilities exactly correctly, so making the distinction does not permit us to improve on the old field. As a result, the best weight we can choose for ‘B’ is 1, which is equivalent to not having the feature ‘B’ at all.

5.3 Selecting the Initial Weight

DD&L show that there is a unique weight $\hat{\beta}$ that maximizes the score for a new feature f (provided that the score for f is not constant for all weights). Writing q_β for the distribution that results from assigning weight β to feature f , $\hat{\beta}$ is the solution to the equation

$$q_\beta[f] = \tilde{p}[f] \tag{10}$$

Intuitively, we choose the weight such that the expectation of f under the resulting new field is equal to its empirical expectation.

Solving equation (10) for β is easy if $L(G)$ is small enough to enumerate. Then the sum over $L(G)$ that is implicit in $q_\beta[f]$ can be expanded out, and solving for β is simply a matter of arithmetic. Things are a bit trickier if $L(G)$ is too large to enumerate. DD&L show that we can solve equation (10) if we can estimate $q_{\text{old}}[f = k]$ for k from 0 to the maximum value of f in the training corpus. (See appendix 1 for details.)

We can estimate $q_{\text{old}}[f = k]$ by means of **random sampling**. The idea is actually rather simple: to estimate how often the feature appears in “the average dag”, we generate a representative mini-corpus from the distribution q_{old} and count. That is, we generate dags at random in such a way that the relative frequency of dag x is $q_{\text{old}}(x)$ (in the limit), and we count how often the feature of interest appears in dags in our generated mini-corpus.

The application that DD&L consider is the induction of English orthographic constraints, that is, inducing a field that assigns high probability to “English-sounding” words and low probability to non-English-sounding words. For this application, Gibbs sampling is appropriate. Gibbs sampling does not work for the application to AV grammars, however. Fortunately, there is an alternative random sampling method we can use: Metropolis-Hastings sampling. We will discuss the issue in some detail shortly.

5.4 Readjusting Weights

When a new feature is added to the field, the best value for its initial weight is chosen, but the weights for the old features are held constant. In general, however, adding the new feature may make it necessary to readjust weights for all features. The second half

of the IIS algorithm involves finding the best weights for a given set of features.

The method is very similar to the method for selecting the initial weight for a new feature. Let $(\beta_1, \dots, \beta_n)$ be the old weights for the features. We wish to compute “increments” $(\delta_1, \dots, \delta_n)$ to determine a new field with weights $(\delta_1\beta_1, \dots, \delta_n\beta_n)$. Consider the equation

$$q_{\text{old}}[\delta_i^{f\#} f_i] = \tilde{p}[f_i] \quad (11)$$

where $f\#(x) = \sum_i f_i(x)$ is the total number of features of dag x . The reason for the factor $\delta_i^{f\#}$ is a bit involved. Very roughly, we would like to choose weights so that the expectation of f_i under the *new* field is equal to $\tilde{p}[f_i]$. Now $q_{\text{new}}(x)$ is:

$$\begin{aligned} q_{\text{new}}(x) &= \frac{1}{Z} p_0(x) \prod_j (\delta_j \beta_j)^{f_j(x)} \\ &= \frac{1}{Z_\delta} q_{\text{old}}(x) \prod_j \delta_j^{f_j(x)} \end{aligned}$$

where we factor Z as $Z_\delta Z_\beta$, for Z_β the normalization constant in q_{old} . Hence, $q_{\text{new}}[f_i] = q_{\text{old}}[\frac{1}{Z_\delta} f_i \prod_j \delta_j^{f_j(x)}]$. Now there are two problems with this expression: it requires us to compute Z_δ , which we are not able to do, and it requires us to determine weights δ_j for all the features simultaneously, not just the weight δ_i for feature i . We might consider approximating $q_{\text{new}}[f_i]$ by ignoring the normalization factor and assuming that all features have the same weight as feature i . Since $\prod_j \delta_i^{f_j(x)} = \delta_i^{f\#(x)}$, we arrive at the expression on the lefthand side of equation (11).

One might expect the approximation just described to be rather poor, but it is proven in (Della Pietra, Della Pietra, and Lafferty, 1995) that solving equation (11) for δ_i (for each i) and setting the new weight for feature i to $\delta_i\beta_i$ is guaranteed to improve the model. This is the real justification for equation (11), and the reader is referred to (Della Pietra, Della Pietra, and Lafferty, 1995) for details.

Solving (11) yields improved weights, but it does not necessarily immediately yield the globally best weights. We can obtain the globally best weights by iterating. Set

$\beta_i \leftarrow \delta_i \beta_i$, for all i , and solve equation (11) again. Repeat until the weights no longer change.

As with equation (10), solving equation (11) is straightforward if $L(G)$ is small enough to enumerate, but not if $L(G)$ is large. In that case, we must use random sampling. We generate a representative mini-corpus and estimate expectations by counting in the mini-corpus. (See appendix 2.)

5.5 Random Sampling

We have seen that random sampling is necessary both to set the initial weight for features under consideration and to adjust all weights after a new feature is adopted. Random sampling involves creating a corpus that is representative of a given model distribution $q(x)$. To take a very simple example, a fair coin can be seen as a method for sampling from the distribution q in which $q(H) = 1/2$, $q(T) = 1/2$. Saying that a corpus is representative is actually not a comment about the corpus itself but the method by which it was generated: a corpus representative of distribution q is one generated by a process that samples from q . Saying that a process M samples from q is to say that the empirical distributions of corpora generated by M converge to q in the limit. For example, if we flip a fair coin once, the resulting empirical distribution over (H, T) is either $(1, 0)$ or $(0, 1)$, not the fair-coin distribution $(1/2, 1/2)$. But as we take larger and larger corpora, the resulting empirical distributions converge to $(1/2, 1/2)$.

An advantage of SCFGs that random fields lack is the transparent relationship between an SCFG defining a distribution q and a sampler for q . We can sample from q by performing stochastic derivations: each time we have a choice among rules expanding a category X , we choose rule $X \rightarrow \xi_i$ with probability β_i , where β_i is the weight of rule $X \rightarrow \xi_i$.

Now we *can* sample from the initial distribution p_0 by performing stochastic deriva-

tions. At the beginning of section 3, we sketched how to generate dags from an AV grammar G via nondeterministic derivations. We defined the initial distribution in terms of weights θ attached to the rules of G . We can convert the nondeterministic derivations discussed at the beginning of section 3 into stochastic derivations by choosing rule $X \rightarrow \xi_i$ with probability θ_i when expanding a node labelled X . Some derivations fail, but throwing away failed derivations has the effect of renormalizing the weight function, so that we generate a dag x with probability $p_0(x)$, as desired.

The Metropolis-Hastings algorithm provides us with a means of converting the sampler for the initial distribution $p_0(x)$ into a sampler for the field distribution $q(x)$. Generally, let $p(\cdot)$ be a distribution for which we have a sampler. We wish to construct a sample x_1, \dots, x_N from a different distribution $q(\cdot)$. Assume that items x_1, \dots, x_n are already in the sample, and we wish to choose x_{n+1} . The sampler for $p(\cdot)$ *proposes* a new item y . We do not simply add y to the sample—that would give us a sample from $p(\cdot)$ —but rather we make a stochastic decision whether to accept the proposal y or reject it. If we accept y , it is added to the sample ($x_{n+1} = y$), and if we reject y , then x_n is repeated ($x_{n+1} = x_n$).

The acceptance decision is made as follows. If $p(y) > q(y)$, then y is overrepresented among the proposals. We can quantify the degree of overrepresentation as $p(y)/q(y)$. The idea is to reject y with a probability corresponding to its degree of overrepresentation. However, we do not consider the absolute degree of overrepresentation, but rather the degree of overrepresentation relative to x_n . (If y and x_n are equally overrepresented, there is no reason to reject y in favor of x_n .) That is, we consider the value

$$r = \frac{p(y)/q(y)}{p(x_n)/q(x_n)} = \frac{p(y)q(x_n)}{p(x_n)q(y)}$$

If $r \leq 1$, then y is underrepresented relative to x_n , and we accept y with probability one.

If $r > 1$, then we accept y with a probability that diminishes as r increases: specifically,

with probability $1/r$. In brief, the acceptance probability of y is $A(y|x_n) = \min(1, 1/r)$.

It can be shown that proposing items with probability $p(\cdot)$ and accepting them with probability $A(\cdot|x_n)$ yields a sampler for $q(\cdot)$. (See e.g. (Winkler, 1995)).²

The acceptance probability $A(y|x_n)$ reduces in our case to a particularly simple form.

If $r < 1$ then $A(y|x) = 1$. Otherwise, writing $\phi(x)$ for the ‘‘field weight’’ $\prod_i \beta_i^{f_i(x)}$, we have:

$$\begin{aligned} A(y|x_n) &= \frac{Z^{-1} \phi(y) p_0(y) p_0(x_n)}{Z^{-1} \phi(x_n) p_0(x_n) p_0(y)} \\ &= \phi(y) / \phi(x_n) \end{aligned} \tag{12}$$

6. Final Remarks

In summary, we cannot simply transplant CF methods to the AV grammar case. In particular, the ERF method yields correct weights only for SCFGs, not for AV grammars.

We can define a probabilistic version of AV grammars with a correct weight-selection method by going to random fields. Feature selection and weight adjustment can be accomplished using the IIS algorithm. In feature selection, we need to use random sampling to find the initial weight for a candidate feature, and in weight adjustment we need to use random sampling to solve the weight equation. The random sampling method that DD&L used is not appropriate for sets of dags, but we can solve that problem by using the Metropolis-Hastings method instead.

² The Metropolis-Hastings acceptance probability is usually given in the form

$$A(y|x) = \min \left(1, \frac{\pi(y)g(y, x)}{\pi(x)g(x, y)} \right)$$

in which π is the distribution we wish to sample from (q , in our notation) and $g(x, y)$ is the proposal probability: the probability that the input sampler will propose y if the previous configuration was x . The case we consider is a special case in which the proposal probability is independent of x : the proposal probability $g(x, y)$ is, in our notation, $p(y)$.

The original Metropolis algorithm is also a special case of the Metropolis-Hastings algorithm, in which the proposal probability is symmetric, that is, $g(x, y) = g(y, x)$. The acceptance function then reduces to $\min(1, \pi(y)/\pi(x))$, which is $\min(1, q(y)/q(x))$ in our notation. I mention this only to point out that it is a *different* special case. Our proposal probability is not symmetric, but rather independent of the previous configuration, and though our acceptance function reduces to a form (12) that is similar to the original Metropolis acceptance function, it is not the same: in general, $\phi(y)/\phi(x) \neq q(y)/q(x)$.

Open questions remain. First, random sampling is notorious for being slow, and it remains to be shown whether the approach proposed here will be practicable. I expect practicability to be quite sensitive to the choice of grammar—the more the grammar’s distribution diverges from the initial context-free approximation, the more features will be necessary to “correct” it, and the more random sampling will be called on.

A second issue is incomplete data. The approach described here assumes complete data (a parsed training corpus). Fortunately, an extension of the method to handle incomplete data (unparsed training corpora) is described in (Riezler, 1997), and I refer readers to that paper.

As a closing note, it should be pointed out explicitly that the random field techniques described here can be profitably applied to context-free grammars, as well. As Stanley Peters nicely put it, there is a distinction between *possibilistic* and *probabilistic* context-sensitivity. Even if the language described by the grammar of interest—that is, the set of possible trees—is context-free, there may well be context-sensitive statistical dependencies. Random fields can be readily applied to capture such statistical dependencies whether or not $L(G)$ is context-sensitive.

Acknowledgments

This work has greatly profited from the comments, criticism, and suggestions of a number of people, including Yoav Freund, John Lafferty, Stanley Peters, Hans Uszkoreit, and members of the audience at talks I gave at Saarbrücken and Tübingen. Michael Miller and Kevin Mark introduced me to random fields as a way of dealing with context-sensitivities in language, planting the idea that led (much later) to this paper. Finally, I would especially like to thank Marc Light and Stefan Riezler for extended discussions of the issues addressed here and helpful criticism of my first attempts to present this material. All responsibility for flaws and errors of course remains with me.

References

- Brew, Chris. 1995. Stochastic HPSG. In *Proceedings of EACL-95*.
- Della Pietra, Stephen, Vincent Della Pietra, and John Lafferty. 1995. Inducing features of random fields. tech report CMU-CS-95-144, CMU.
- Eisele, Andreas. 1994. Towards probabilistic extensions of constraint-based grammars. Technical Report Deliverable R1.2.B, DYANA-2.
- Gibbs, W. 1902. *Elementary principles of statistical mechanics*. Yale University Press, New Haven, CT.
- Mark, Kevin, Michael Miller, Ulf Grenander, and Steve Abney. 1992. Parameter estimation for constrained context-free language models. In *Proceedings of the Fifth Darpa Workshop on Speech and Natural Language*, San Mateo, CA. Morgan Kaufman.
- Riezler, Stefan. 1996. Quantitative constraint logic programming for

weighted grammar applications. Talk given at LACL, September.

Riezler, Stefan. 1997. Probabilistic Constraint Logic Programming. Arbeitspapiere des Sonderforschungsbereichs 340, Bericht Nr. 117, Universität Tübingen.

Winkler, Gerhard. 1995. *Image Analysis, Random Fields and Dynamic Monte Carlo Methods*. Springer.

A. Initial Weight Estimation

In the feature selection step, we choose an initial weight β for each candidate feature f so as to maximize the *gain* $G = D(\tilde{p} \| q_{\text{old}}) - D(\tilde{p} \| q_{f,\beta})$ of adding f to the field. It is actually more convenient to consider log weights $\alpha = \ln \beta$. For a given feature f , the log weight $\hat{\alpha}$ that maximizes gain is the solution to the equation:

$$q_\alpha[f] = \tilde{p}[f]$$

where q_α is the distribution that results from adding f to the field with log weight α . This equation can be solved using Newton's method. Define

$$F(\alpha) = \tilde{p}[f] - q_\alpha[f] \quad (13)$$

To find the value of α for which $F(\alpha) = 0$, we begin at a convenient point α_0 (the “null” weight $\alpha_0 = 0$ recommends itself)

and iteratively compute:

$$\alpha_{t+1} = \alpha_t - \frac{F(\alpha_t)}{F'(\alpha_t)} \quad (14)$$

(Della Pietra, Della Pietra, and Lafferty, 1995) show that $F'(\alpha_t)$ is equal to the negative of the variance of f under the new field, which I will write $-V_\alpha[f]$.

To compute the iteration (14) we need to be able to compute $F(\alpha_t)$ and $F'(\alpha_t)$. For $F(\alpha_t)$ we require $\tilde{p}[f]$ and $q_\alpha[f]$, and $F'(\alpha_t)$ can be expressed as $q_\alpha[f]^2 - q_\alpha[f^2]$. $\tilde{p}[f]$ is simply the average value of f in the training corpus. The remaining terms are all of the form $q_\alpha[f^r]$. We can re-express this expectation in terms of the old field

q_{old} :

$$\begin{aligned} q_\alpha[f^r] &= \frac{\sum_x f^r(x) q_\alpha(x)}{\sum_x f^r(x) e^{\alpha f(x)} q_{\text{old}}(x)} \\ &= \frac{\sum_x e^{\alpha f(x)} q_{\text{old}}(x)}{\sum_x e^{\alpha f(x)} q_{\text{old}}(x)} \\ &= \frac{q_{\text{old}}[f^r e^{\alpha f}]}{q_{\text{old}}[e^{\alpha f}]} \end{aligned}$$

The expectations $q_{\text{old}}[f^r e^{\alpha f}]$ can be ob-

tained by generating a random sample (z_1, \dots, z_N)

of size N from q_{old} and computing the av-

erage value of $f^r e^{\alpha f}$. That is, $q_{\text{old}}[f^r e^{\alpha f}] \approx$

$(1/N) s_r(\alpha)$, where:

$$\begin{aligned} s_r(\alpha) &= \sum_k f^r(z_k) e^{\alpha f(z_k)} \\ &= \sum_u \text{count}_k[f(z_k) = u] u^r e^{\alpha u} \end{aligned}$$

This yields:

$$q_\alpha[f^r] = \frac{s_r(\alpha)}{s_0(\alpha)}$$

and the Newton iteration (14) reduces to:

$$\alpha_{t+1} = \alpha_t + \frac{s_0^2(\alpha_t)\tilde{p}[f] - s_0(\alpha_t)s_1(\alpha_t)}{s_0(\alpha_t)s_2(\alpha_t) - s_1(\alpha_t)^2}$$

To compare candidates, we also need to know the gain $D(\tilde{p}\|q_{\text{old}}) - D(\tilde{p}\|q_{\hat{\alpha}})$ for each candidate. This can be expressed as follows (Della Pietra, Della Pietra, and Lafferty, 1995):

$$\begin{aligned} G(f, \hat{\alpha}) &= \tilde{p}[f] \ln \hat{\alpha} - \ln q_{\text{old}}[e^{\hat{\alpha}f}] \\ &\approx \tilde{p}[f] \ln \hat{\alpha} - \ln s_0(\hat{\alpha}) + \ln N \end{aligned}$$

Putting everything together, the algorithm for feature selection has the following form. The array $E[f]$ is assumed to have been initialized with the empirical expectations $\tilde{p}[f]$.

procedure *SelectFeature* () **begin**

Fill array $C[f, u] = \text{count}_k[f(z_k) = u]$

by sampling from old field

$\hat{G} \leftarrow 0, g \leftarrow \text{none}$

for each f **in candidates** **do**

$\alpha \leftarrow 0$

until α *is accurate enough* **do**

$s_0 \leftarrow s_1 \leftarrow s_2 \leftarrow 0$

for u **from** 0 **to** u_{max} **do**

$x \leftarrow C[f, u]e^{\alpha u}$

$s_0 \leftarrow s_0 + x$

$s_1 \leftarrow s_0 + xu$

$s_2 \leftarrow s_0 + xu^2$

end

$\alpha \leftarrow \alpha + \frac{s_0^2 E[f] - s_0 s_1}{s_0 s_2 - s_1^2}$

end

$G \leftarrow \alpha E[f] - \ln s_0 + \ln N$

if $G > \hat{G}$ **then** $\hat{G} \leftarrow G, g \leftarrow f, \hat{\alpha} \leftarrow \alpha$

end

return $g, \hat{\alpha}, \hat{G}$

end

B. Adjusting Field Weights

The procedure for adjusting field weights has much the same structure as the procedure for choosing initial weights. In terms of log weights, we wish to compute increments $(\delta_1, \dots, \delta_n)$ such that the new field, with log weights $(\alpha_1 + \delta_1, \dots, \alpha_n + \delta_n)$ has a lower divergence than the old field $(\alpha_1, \dots, \alpha_n)$. We choose each δ_i as the solution to the equation:

$$\tilde{p}[f_i] = q_{\text{old}}[f_i e^{\delta_i f_i \#}]$$

Again, we use Newton's method. We wish to find δ such that $F_i(\delta) = 0$, where:

$$F_i(\delta) = \tilde{p}[f_i] - q_{\text{old}}[f_i e^{\delta f_{\#}}]$$

As (Della Pietra, Della Pietra, and Laferty, 1995) show, the first derivative is:

$$F'_i(\delta) = -q_{\text{old}}[f_i f_{\#} e^{\delta f_{\#}}]$$

We see that the expectations we need to compute by sampling from q_{old} are of form $q_{\text{old}}[f_i f_{\#}^r e^{\delta f_{\#}}]$. We generate a random sample (z_1, \dots, z_N) and define:

$$\begin{aligned} s_r(i, \delta) &= \sum_k f_i(z_k) f_{\#}(z_k)^r e^{\delta f_{\#}(z_k)} \\ &= \sum_m \sum_u \text{count}_k[f_i(z_k) = u \wedge f_{\#}(z_k) = m] u m^r e^{\delta m} \\ &= \sum_m m^r e^{\delta m} \sum_{k|f_{\#}(z_k)=m} f_i(z_k) \end{aligned}$$

As we generate the sample we update the array $C[i, m] = \sum_{k|f_{\#}(z_k)=m} f_i(z_k)$. We estimate $q_{\text{old}}[f_i f_{\#}^r e^{\delta f_{\#}}]$ as the average value of $f_i f_{\#}^r e^{\delta f_{\#}}$ in the sample, namely, $(1/N)s_r(i, \delta)$.

This permits us to compute $F_i(\delta)$ and $F'_i(\delta)$.

The resulting Newton iteration is:

$$\delta_{t+1} = \delta_t + \frac{N\tilde{p}[f_i] - s_0(i, \delta_t)}{s_1(i, \delta)}$$

The estimation procedure is:

```

procedure AdjustWeights ( $\alpha_1, \dots, \alpha_n$ ) begin
  until the field converges do
    Fill array  $C[i, m]$ 

```

by sampling from q_{α}

for i **from** 1 **to** n

$\delta \leftarrow 0$

until δ is sufficiently accurate **do**

$s_0 \leftarrow s_1 \leftarrow 0$

for m **from** 0 **to** m_{max} **do**

$x \leftarrow C[i, m] e^{\delta m}$

$s_0 \leftarrow s_0 + x$

$s_1 \leftarrow s_1 + x m$

end

$\delta \leftarrow \delta + \frac{NE[f_i] - s_0}{s_1}$

end

end

return $(\alpha_1, \dots, \alpha_n)$

end

A Maximum Entropy Approach to Natural Language Processing

Adam L. Berger*
Stephen A. Della Pietra†
Vincent J. Della Pietra†

IBM T.J. Watson Research Center¹
P.O. Box 704
Yorktown Heights, NY 10598

The concept of maximum entropy can be traced back along multiple threads to Biblical times. Only recently, however, have computers become powerful enough to permit the widescale application of this concept to real world problems in statistical estimation and pattern recognition. In this paper we describe a method for statistical modeling based on maximum entropy. We present a maximum-likelihood approach for automatically constructing maximum entropy models and describe how to implement this approach efficiently, using as examples several problems in natural language processing.

1. Introduction

Statistical modeling addresses the problem of constructing a stochastic model to predict the behavior of a random process. In constructing this model, we typically have at our disposal a sample of output from the process. Given this sample, representing an incomplete state of knowledge about the process, the modeling problem is to parlay this knowledge into a representation of the process. We can then use this representation to make predictions of the future behavior of the process.

Baseball managers (who rank among the better paid statistical modelers) employ batting averages, compiled from a history of at-bats, to gauge the likelihood that a player will succeed in his next appearance at the plate. Thus informed, they manipulate their lineups accordingly. Wall Street speculators (who rank among the *best* paid statistical modelers) build models based on past stock price movements to predict tomorrow's fluctuations and alter their portfolios to capitalize on the predicted future. At the other end of the pay scale reside natural language researchers, who design language and acoustic models for use in speech recognition systems and related applications.

The past few decades have witnessed significant progress toward increasing the predictive capacity of statistical models of natural language. In language modeling, for instance, (Bahl *et al* 1989) have used decision tree models and (Della Pietra *et al* 1994) have used automatically inferred link grammars to model long range correlations in language. In parsing, (Black *et al* 1992) has described how to extract grammatical rules from

* Now at Columbia University computer science department

† Now at Renaissance Technologies, Stony Brook, NY

¹ Research supported in part by ARPA under grant ONR N00014-91-C-0135

annotated text automatically and incorporate these rules into statistical models of grammar. In speech recognition, (Lucassen and Mercer 1984) have introduced a technique for automatically discovering relevant features for the translation of word spelling to word pronunciation.

These efforts, while varied in specifics, all confront two essential tasks of statistical modeling. The first task is to determine a set of statistics which capture the behavior of a random process. Given a set of statistics, the second task is to corral these facts into an accurate model of the process—a model capable of predicting the future output of the process. The first task is one of feature selection; the second is one of model selection. In the following pages we present a unified approach to these two tasks based on the maximum entropy philosophy.

Our discussion will proceed as follows. In Section 2 we give an overview of the maximum entropy philosophy and work through a motivating example. In Section 3 we describe the mathematical structure of maximum entropy models and give an efficient algorithm for estimating the parameters of such models. In Section 4 we discuss feature selection, and present an automatic method for discovering facts about a process from a sample of output from the process. We then present a series of refinements to the method to make it practical to implement. Finally, in Section 5 we describe the application of maximum entropy ideas to several tasks in stochastic language processing: bilingual sense disambiguation, word reordering, and sentence segmentation.

2. A Maximum Entropy Overview

We introduce the concept of maximum entropy through a simple example. Suppose we wish to model an expert translator's decisions concerning the proper French rendering of the English word *in*. Our model p of the expert's decisions assigns to each French word or phrase f an estimate, $p(f)$, of the probability that the expert would choose f as a translation of *in*. To guide us in developing p , we collect a large sample of instances of the expert's decisions. Our goal is to extract a set of facts about the decision-making process from the sample (the first task of modeling) that will aid us in constructing a model of this process (the second task).

One obvious clue we might glean from the sample is the list of allowed translations. For example, we might discover that the expert translator always chooses among the following five French phrases: $\{dans, en, \grave{a}, au\ cours\ de, pendant\}$. With this information in hand, we can impose our first constraint on our model p :

$$p(dans) + p(en) + p(\grave{a}) + p(au\ cours\ de) + p(pendant) = 1$$

This equation represents our first statistic of the process; we can now proceed to search for a suitable model which obeys this equation. Of course, there are an infinite number of models p for which this identity holds. One model which satisfies the above equation is $p(dans) = 1$; in other words, the model always predicts *dans*. Another model which obeys this constraint predicts *pendant* with a probability of $1/2$, and *\grave{a}* with a probability of $1/2$. But both of these models offend our sensibilities: knowing only that the expert always chose from among these five French phrases, how can we justify either of these probability distributions? Each seems to be making rather bold assumptions, with no empirical justification. Put another way, these two models assume more than we actually know about the expert's decision-making process. All we know is that the expert chose exclusively from among these five French phrases; given this, the most intuitively appealing model is the following:

$$\begin{aligned}
p(\textit{dans}) &= 1/5 \\
p(\textit{en}) &= 1/5 \\
p(\grave{\textit{a}}) &= 1/5 \\
p(\textit{au cours de}) &= 1/5 \\
p(\textit{pendant}) &= 1/5
\end{aligned}$$

This model, which allocates the total probability evenly among the five possible phrases, is the most uniform model subject to our knowledge. It is not, however, the most uniform overall; that model would grant an equal probability to every *possible* French phrase.

We might hope to glean more clues about the expert's decisions from our sample. Suppose we notice that the expert chose either *dans* or *en* 30% of the time. We could apply this knowledge to update our model of the translation process by requiring that p satisfy two constraints:

$$\begin{aligned}
p(\textit{dans}) + p(\textit{en}) &= 3/10 \\
p(\textit{dans}) + p(\textit{en}) + p(\grave{\textit{a}}) + p(\textit{au cours de}) + p(\textit{pendant}) &= 1
\end{aligned}$$

Once again there are many probability distributions consistent with these two constraints. In the absence of any other knowledge, a reasonable choice for p is again the most uniform—that is, the distribution which allocates its probability as evenly as possible, subject to the constraints:

$$\begin{aligned}
p(\textit{dans}) &= 3/20 \\
p(\textit{en}) &= 3/20 \\
p(\grave{\textit{a}}) &= 7/30 \\
p(\textit{au cours de}) &= 7/30 \\
p(\textit{pendant}) &= 7/30
\end{aligned}$$

Say we inspect the data once more, and this time notice another interesting fact: in half the cases, the expert chose either *dans* or *à*. We can incorporate this information into our model as a third constraint:

$$\begin{aligned}
p(\textit{dans}) + p(\textit{en}) &= 3/10 \\
p(\textit{dans}) + p(\textit{en}) + p(\grave{\textit{a}}) + p(\textit{au cours de}) + p(\textit{pendant}) &= 1 \\
p(\textit{dans}) + p(\grave{\textit{a}}) &= 1/2
\end{aligned}$$

We can once again look for the most uniform p satisfying these constraints, but now the choice is not as obvious. As we have added complexity, we have encountered two difficulties at once. First, what exactly is meant by “uniform,” and how can one measure the uniformity of a model? Second, having determined a suitable answer to these questions, how does one go about finding the most uniform model subject to a set of constraints like those we have described?

The maximum entropy method answers both these questions, as we will demonstrate in the next few pages. Intuitively, the principle is simple: model all that is known and

assume nothing about that which is unknown. In other words, given a collection of facts, choose a model which is consistent with all the facts, but otherwise as uniform as possible. This is precisely the approach we took in selecting our model p at each step in the above example.

The maximum entropy concept has a long history. Adopting the least complex hypothesis possible is embodied in Occam’s Razor (“Nunquam ponenda est pluralitas sine necessitate”) and even appears earlier, in the Bible and the writings of Herotodus (Jaynes 1990). Laplace might justly be considered the father of maximum entropy, having enunciated the underlying theme 200 years ago in his “Principle of Insufficient Reason”: when one has no information to distinguish between the probability of two events, the best strategy is to consider them equally likely (Guisasu and Shenitzer 1994). As E.T. Jaynes, a more recent pioneer of maximum entropy, put it (Jaynes 1990):

...the fact that a certain probability distribution maximizes entropy subject to certain constraints representing our incomplete information, is the fundamental property which justifies use of that distribution for inference; it agrees with everything that is known, but carefully avoids assuming anything that is not known. It is a transcription into mathematics of an ancient principle of wisdom...

3. Maximum Entropy Modeling

We consider a random process which produces an output value y , a member of a finite set \mathcal{Y} . For the translation example just considered, the process generates a translation of the word *in*, and the output y can be any word in the set $\{dans, en, \grave{a}, au\ cours\ de, pendant\}$. In generating y , the process may be influenced by some contextual information x , a member of a finite set \mathcal{X} . In the present example, this information could include the words in the English sentence surrounding *in*.

Our task is to construct a stochastic model that accurately represents the behavior of the random process. Such a model is a method of estimating the conditional probability that, given a context x , the process will output y . We will denote by $p(y|x)$ the probability that the model assigns to y in context x . With a slight abuse of notation, we will also use $p(y|x)$ to denote the entire conditional probability distribution provided by the model, with the interpretation that y and x are placeholders rather than specific instantiations. The proper interpretation should be clear from the context. We will denote by \mathcal{P} the set of all conditional probability distributions. Thus a model $p(y|x)$ is, by definition, just an element of \mathcal{P} .

3.1 Training Data

To study the process, we observe the behavior of the random process for some time, collecting a large number of samples $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$. In the example we have been considering, each sample would consist of a phrase x containing the words surrounding *in*, together with the translation y of *in* which the process produced. For now we can imagine that these training samples have been generated by a human expert who was presented with a number of random phrases containing *in* and asked to choose a good translation for each. When we discuss real-world applications in Section 5, we will show how such samples can be automatically extracted from a bilingual corpus.

We can summarize the training sample in terms of its empirical probability distri-

bution \tilde{p} , defined by

$$\tilde{p}(x, y) \equiv \frac{1}{N} \times \text{number of times that } (x, y) \text{ occurs in the sample}$$

Typically, a particular pair (x, y) will either not occur at all in the sample, or will occur at most a few times.

3.2 Statistics, Features and Constraints

Our goal is to construct a statistical model of the process which generated the training sample $\tilde{p}(x, y)$. The building blocks of this model will be a set of statistics of the training sample. In the current example we have employed several such statistics: the frequency that *in* translated to either *dans* or *en* was 3/10; the frequency that it translated to either *dans* or *au cours de* was 1/2; and so on. These particular statistics were independent of the context, but we could also consider statistics which depend on the conditioning information x . For instance, we might notice that, in the training sample, if *April* is the word following *in*, then the translation of *in* is *en* with frequency 9/10.

To express the event that *in* translates as *en* when *April* is the following word, we can introduce the indicator function

$$f(x, y) = \begin{cases} 1 & \text{if } y = \textit{en} \text{ and } \textit{April} \text{ follows } \textit{in} \\ 0 & \text{otherwise} \end{cases}$$

The expected value of f with respect to the empirical distribution $\tilde{p}(x, y)$ is exactly the statistic we are interested in. We denote this expected value by

$$\tilde{p}(f) \equiv \sum_{x,y} \tilde{p}(x, y) f(x, y) \quad (1)$$

We can express any statistic of the sample as the expected value of an appropriate binary-valued indicator function f . We call such function a *feature function* or *feature* for short. (As with probability distributions, we will sometimes abuse notation and use $f(x, y)$ to denote both the value of f at a particular pair (x, y) as well as the entire function f .)

When we discover a statistic that we feel is useful, we can acknowledge its importance by requiring that our model accord with it. We do this by constraining the expected value that the model assigns to the corresponding feature function f . The expected value of f with respect to the model $p(y|x)$ is

$$p(f) \equiv \sum_{x,y} \tilde{p}(x) p(y|x) f(x, y) \quad (2)$$

where $\tilde{p}(x)$ is the empirical distribution of x in the training sample. We constrain this expected value to be the same as the expected value of f in the training sample. That is, we require

$$p(f) = \tilde{p}(f) \quad (3)$$

Combining (1), (2) and (3) yields the more explicit equation

$$\sum_{x,y} \tilde{p}(x) p(y|x) f(x, y) = \sum_{x,y} \tilde{p}(x, y) f(x, y)$$

We call the requirement (3) a *constraint equation* or simply a *constraint*. By restricting attention to those models $p(y|x)$ for which (3) holds, we are eliminating from consideration those models which do not agree with the training sample on how often the output of the process should exhibit the feature f .

To sum up so far, we now have a means of representing statistical phenomena inherent in a sample of data (namely, $\tilde{p}(f)$), and also a means of requiring that our model of the process exhibit these phenomena (namely, $p(f) = \tilde{p}(f)$).

One final note about features and constraints bears repeating: though the words “feature” and “constraint” are often used interchangeably in discussions of maximum entropy, we will be vigilant to distinguish the two and urge the reader to do likewise: a feature is a binary-valued function of (x, y) ; a constraint is an equation between the expected value of the feature function in the model and its expected value in the training data.

3.3 The Maximum Entropy Principle

Suppose that we are given n feature functions f_i , which determine statistics we feel are important in modeling the process. We would like our model to accord with these statistics. That is, we would like p to lie in the subset \mathcal{C} of \mathcal{P} defined by

$$\mathcal{C} \equiv \left\{ p \in \mathcal{P} \mid p(f_i) = \tilde{p}(f_i) \quad \text{for } i \in \{1, 2, \dots, n\} \right\} \quad (4)$$

Figure 1 provides a geometric interpretation of this setup. Here \mathcal{P} is the space of all (unconditional) probability distributions on 3 points, sometimes called a *simplex*. If we impose no constraints (depicted in (a)), then all probability models are allowable. Imposing one linear constraint \mathcal{C}_1 restricts us to those $p \in \mathcal{P}$ which lie on the region defined by \mathcal{C}_1 , as shown in (b). A second linear constraint could determine p exactly, if the two constraints are satisfiable; this is the case in (c), where the intersection of \mathcal{C}_1 and \mathcal{C}_2 is non-empty. Alternatively, a second linear constraint could be inconsistent with the first—for instance, the first might require that the probability of the first point is $1/3$ and the second that the probability of the third point is $3/4$ —this is shown in (d). In the present setting, however, the linear constraints are extracted from the training sample and cannot, by construction, be inconsistent. Furthermore, the linear constraints in our applications will not even come close to determining $p \in \mathcal{P}$ uniquely as they do in (c); instead, the set $\mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2 \cap \dots \cap \mathcal{C}_n$ of allowable models will be infinite.

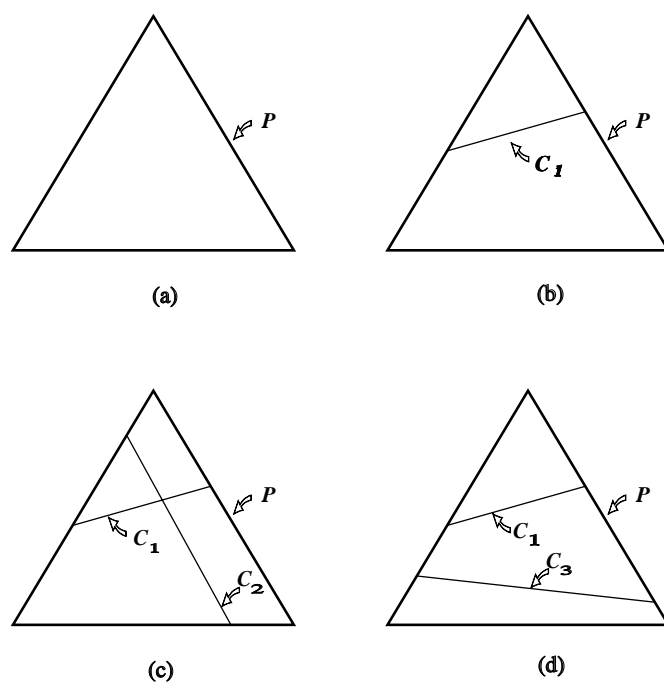
Among the models $p \in \mathcal{C}$, the maximum entropy philosophy dictates that we select the distribution which is most uniform. But now we face a question left open in Section 2: what does “uniform” mean?

A mathematical measure of the uniformity of a conditional distribution $p(y|x)$ is provided by the conditional entropy²

$$H(p) \equiv - \sum_{x,y} \tilde{p}(x)p(y|x) \log p(y|x) \quad (5)$$

The entropy is bounded from below by zero, the entropy of a model with no uncertainty at all, and from above by $\log |\mathcal{Y}|$, the entropy of the uniform distribution over all possible

² A more common notation for the conditional entropy is $H(Y | X)$, where Y and X are random variables with joint distribution $\tilde{p}(x)p(y|x)$. To emphasize the dependence of the entropy on the probability distribution p , we have adopted the alternate notation $H(p)$.

**Figure 1**

Four different scenarios in constrained optimization. \mathcal{P} represents the space of all probability distributions. In (a), no constraints are applied, and all $p \in \mathcal{P}$ are allowable. In (b), the constraint \mathcal{C}_1 narrows the set of allowable models to those which lie on the line defined by the linear constraint. In (c), two consistent constraints \mathcal{C}_1 and \mathcal{C}_2 define a single model $p \in \mathcal{C}_1 \cap \mathcal{C}_2$. In (d), the two constraints are inconsistent (i.e. $\mathcal{C}_1 \cap \mathcal{C}_3 = \emptyset$); no $p \in \mathcal{P}$ can satisfy them both.

$|\mathcal{Y}|$ values of y . With this definition in hand, we are ready to present the principle of maximum entropy.

To select a model from a set \mathcal{C} of allowed probability distributions, choose the model $p_\star \in \mathcal{C}$ with maximum entropy $H(p)$:

$$p_\star = \operatorname{argmax}_{p \in \mathcal{C}} H(p) \quad (6)$$

It can be shown that p_\star is always well-defined; that is, there is always a unique model p_\star with maximum entropy in any constrained set \mathcal{C} .

3.4 Parametric Form

The maximum entropy principle presents us with a problem in constrained optimization: find the $p_\star \in \mathcal{C}$ which maximizes $H(p)$. In simple cases, we can find the solution to this problem analytically. This was true for the example presented in Section 2 when we imposed the first two constraints on p . Unfortunately, the solution of the general maximum entropy cannot be written explicitly, and we need a more indirect approach. (The reader is invited to try to calculate the solution for the same example when the third constraint is imposed.)

To address the general problem, we apply the method of Lagrange multipliers from the theory of constrained optimization. The relevant steps are outlined here; the reader is referred to (Della Pietra *et al* 1995) for a more thorough discussion of constrained optimization as applied to maximum entropy.

- We will refer to the original constrained optimization problem,

$$\text{find } p_\star = \operatorname{argmax}_{p \in \mathcal{C}} H(p)$$

as the *primal* problem.

- For each feature f_i we introduce a parameter λ_i (a Lagrange multiplier). We define the Lagrangian $\Lambda(p, \lambda)$ by

$$\Lambda(p, \lambda) \equiv H(p) + \sum_i \lambda_i (p(f_i) - \tilde{p}(f_i)) \quad (7)$$

- Holding λ fixed, we compute the unconstrained maximum of the Lagrangian $\Lambda(p, \lambda)$ over all $p \in \mathcal{P}$. We denote by p_λ the p where $\Lambda(p, \lambda)$ achieves its maximum and by $\Psi(\lambda)$ the value at this maximum:

$$p_\lambda \equiv \operatorname{argmax}_{p \in \mathcal{P}} \Lambda(p, \lambda) \quad (8)$$

$$\Psi(\lambda) \equiv \Lambda(p_\lambda, \lambda) \quad (9)$$

We call $\Psi(\lambda)$ the *dual* function. The functions p_λ and $\Psi(\lambda)$ may be calculated explicitly using simple calculus. We find

$$p_\lambda(y|x) = \frac{1}{Z_\lambda(x)} \exp \left(\sum_i \lambda_i f_i(x, y) \right) \quad (10)$$

$$\Psi(\lambda) = - \sum_x \tilde{p}(x) \log Z_\lambda(x) + \sum_i \lambda_i \tilde{p}(f_i) \quad (11)$$

where $Z_\lambda(x)$ is a normalizing constant determined by the requirement that $\sum_y p_\lambda(y|x) = 1$ for all x :

$$Z_\lambda(x) = \sum_y \exp \left(\sum_i \lambda_i f_i(x, y) \right) \quad (12)$$

- Finally, we pose the unconstrained *dual* optimization problem

$$\text{Find } \lambda^* = \underset{\lambda}{\operatorname{argmax}} \Psi(\lambda)$$

At first glance it is not clear what these machinations achieve. However, a fundamental principle in the theory of Lagrange multipliers, called generically the Kuhn-Tucker theorem, asserts that under suitable assumptions, the primal and dual problems are, in fact, closely related. This is the case in the present situation. Although a detailed account of this relationship is beyond the scope of this paper, it is easy to state the final result: Suppose that λ^* is the solution of the dual problem. Then p_{λ^*} is the solution of the primal problem; that is $p_{\lambda^*} = p_*$. In other words,

The maximum entropy model subject to the constraints \mathcal{C} has the parametric form³ p_{λ^*} of (10), where the parameter values λ^* can be determined by maximizing the dual function $\Psi(\lambda)$.

The most important practical consequence of this result is that any algorithm for finding the maximum λ^* of $\Psi(\lambda)$ can be used to find the maximum p_* of $H(p)$ for $p \in \mathcal{C}$.

3.5 Relation to Maximum Likelihood

The log-likelihood $L_{\tilde{p}}(p)$ of the empirical distribution \tilde{p} as predicted by a model p is defined by⁴

$$L_{\tilde{p}}(p) \equiv \log \prod_{x,y} p(y|x)^{\tilde{p}(x,y)} = \sum_{x,y} \tilde{p}(x,y) \log p(y|x) \quad (13)$$

It is easy to check that the dual function $\Psi(\lambda)$ of the previous section is, in fact, just the log-likelihood for the exponential model p_λ ; that is

$$\Psi(\lambda) = L_{\tilde{p}}(p_\lambda) \quad (14)$$

With this interpretation, the result of the previous section can be rephrased as:

The model $p_* \in \mathcal{C}$ with maximum entropy is the model in the parametric family $p_\lambda(y|x)$ that maximizes the likelihood of the training sample \tilde{p} .

³ It might be that the dual function $\Psi(\lambda)$ does not achieve its maximum at any finite λ^* . In this case, the maximum entropy model will not have the form p_λ for any λ . However, it will be the *limit* of models of this form, as indicated by the following result whose proof we omit:

Suppose λ_n is any sequence such that $\Psi(\lambda_n)$ converges to the maximum of $\Psi(\lambda)$. Then p_{λ_n} converges to p_* .

⁴ We will henceforth abbreviate $L_{\tilde{p}}(p)$ by $L(p)$ when the empirical distribution \tilde{p} is clear from context.

	Primal	Dual
<i>problem</i>	$\operatorname{argmax}_{p \in \mathcal{C}} H(p)$	$\operatorname{argmax}_{\lambda} \Psi(\lambda)$
<i>description</i>	maximum entropy	maximum likelihood
<i>type of search</i>	constrained optimization	unconstrained optimization
<i>search domain</i>	$p \in \mathcal{C}$	real-valued vectors $\{\lambda_1, \lambda_2 \dots\}$
<i>solution</i>	p_*	λ^*
Kuhn-Tucker theorem: $p_* = p_{\lambda^*}$		

Table 1

The duality of maximum entropy and maximum likelihood is an example of the more general phenomenon of duality in constrained optimization.

This result provides an added justification for the maximum entropy principle: if the notion of selecting a model p_* on the basis of maximum entropy isn't compelling enough, it so happens that this same p_* is also the model which, from among all models of the same parametric form (10), can best account for the training sample.

Table 1 summarizes the primal-dual framework we have established.

3.6 Computing the Parameters

For all but the most simple problems, the λ^* that maximize $\Psi(\lambda)$ cannot be found analytically. Instead, we must resort to numerical methods. From the perspective of numerical optimization, the function $\Psi(\lambda)$ is well behaved, since it is smooth and convex- \cap in λ . Consequently, a variety of numerical methods can be used to calculate λ^* . One simple method is coordinate-wise ascent, in which λ^* is computed by iteratively maximizing $\Psi(\lambda)$ one coordinate at a time. When applied to the maximum entropy problem, this technique yields the popular Brown algorithm (Brown 1959). Other general purpose methods that can be used to maximize $\Psi(\lambda)$ include gradient ascent and conjugate gradient.

An optimization method specifically tailored to the maximum entropy problem is the iterative scaling algorithm of Darroch and Ratcliff (Darroch and Ratcliff 1972). We present here a version of this algorithm specifically designed for the problem at hand; a proof of the monotonicity and convergence of the algorithm is given in (Della Pietra *et al* 1995). The algorithm is applicable whenever the feature functions $f_i(x, y)$ are non-negative:

$$f_i(x, y) \geq 0 \quad \text{for all } i, x, \text{ and } y \quad (15)$$

This is, of course, true for the binary-valued feature functions we are considering here. The algorithm generalizes the Darroch-Ratcliff procedure, which requires, in addition to the non-negativity, that the feature functions satisfy $\sum_i f_i(x, y) = 1$ for all x, y .

Algorithm 1 : Improved Iterative Scaling

Input: Feature functions f_1, f_2, \dots, f_n ; empirical distribution $\tilde{p}(x, y)$
Output: Optimal parameter values λ^*_i ; optimal model p_{λ^*}

1. Start with $\lambda_i = 0$ for all $i \in \{1, 2, \dots, n\}$
2. Do for each $i \in \{1, 2, \dots, n\}$:
 - a. Let $\Delta\lambda_i$ be the solution to

$$\sum_{x, y} \tilde{p}(x)p(y|x)f_i(x, y)\exp(\Delta\lambda_i f_i^\#(x, y)) = \tilde{p}(f_i) \quad (16)$$

$$\text{where } f^\#(x, y) \equiv \sum_{i=1}^n f_i(x, y) \quad (17)$$

- b. Update the value of λ_i according to: $\lambda_i \leftarrow \lambda_i + \Delta\lambda_i$
3. Go to step 2 if not all the λ_i have converged

The key step in the algorithm is step (2a), the computation of the increments $\Delta\lambda_i$ that solve (16). If $f^\#(x, y)$ is constant ($f^\#(x, y) = M$ for all x, y , say) then $\Delta\lambda_i$ is given explicitly as

$$\Delta\lambda_i = \frac{1}{M} \log \frac{\tilde{p}(f_i)}{p_{\lambda}(f_i)}$$

If $f^\#(x, y)$ is not constant, then $\Delta\lambda_i$ must be computed numerically. A simple and effective way of doing this is by Newton's method. This method computes the solution α_* of an equation $g(\alpha_*) = 0$ iteratively by the recurrence

$$\alpha_{n+1} = \alpha_n - \frac{g(\alpha_n)}{g'(\alpha_n)} \quad (18)$$

with an appropriate choice for α_0 and suitable attention paid to the domain of g .

4. Feature Selection

Earlier we divided the statistical modeling problem into two steps: finding appropriate facts about the data; the second is to incorporate these facts into the model. Up to this point we have proceeded by assuming that the first task was somehow performed for us. Even in the simple example of Section 2, we did not explicitly state how we selected those particular constraints. That is, why is the fact that *dans* or *à* was chosen by the expert translator 50% of the time any more important than countless other facts contained in the data? In fact, the principle of maximum entropy does not directly concern itself with the issue of feature selection: it merely provides a recipe for combining constraints into a model. But the feature selection problem is critical, since the universe of possible constraints is typically in the thousands or even millions. In this section we introduce a method for automatically selecting the features to be included in a maximum entropy model, and then offer a series of refinements to ease the computational burden.

4.1 Motivation

We begin by specifying a large collection \mathcal{F} of candidate features. We do not require *a priori* that these features are actually relevant or useful. Instead, we let the pool be as large as practically possible. Only a small subset of this collection of features will eventually be employed in our final model.

If we had a training sample of infinite size, we could determine the "true" expected value for a candidate feature $f \in \mathcal{F}$ simply by computing the fraction of events in the sample for which $f(x, y) = 1$. In real-life applications, however, we are provided with only a small sample of N events, which cannot be trusted to represent the process fully and accurately. Specifically, we cannot expect that for every feature $f \in \mathcal{F}$, the estimate of $\tilde{p}(f)$ we derive from this sample will be close to its value in the limit as n grows large. Employing a larger (or even just a different) sample of data from the same process might result in different estimates of $\tilde{p}(f)$ for many candidate features.

In short, we would like to include in the model only a subset \mathcal{S} of the full set of candidate features \mathcal{F} . We will call \mathcal{S} the set of *active* features. The choice of \mathcal{S} must capture as much information about the random process as possible, yet only include features whose expected values can be reliably estimated.

To find \mathcal{S} , we adopt an incremental approach to feature selection, similar to the strategy used for growing decision trees (Bahl *et al* 1989). The idea is to build up \mathcal{S} by successively adding features. The choice of feature to add at each step is determined by the training data. Let us denote the set of models determined by the feature set \mathcal{S} as $\mathcal{C}(\mathcal{S})$. “Adding” a feature f is shorthand for requiring that the set of allowable models all satisfy the equality $\hat{p}(f) = p(f)$. Only some members of $\mathcal{C}(\mathcal{S})$ will satisfy this equality; the ones that do we denote by $\mathcal{C}(\mathcal{S} \cup f)$.

Thus, each time a candidate feature is adjoined to \mathcal{S} , another linear constraint is imposed on the space $\mathcal{C}(\mathcal{S})$ of models allowed by the features in \mathcal{S} . As a result, $\mathcal{C}(\mathcal{S})$ shrinks; the model p_* in \mathcal{C} with the greatest entropy reflects ever-increasing knowledge and thus, hopefully, becomes a more accurate representation of the process. This narrowing of the space of permissible models was represented in figure 1 by a series of intersecting lines (hyperplanes, in general) in a probability simplex. Perhaps more intuitively, we could represent it by a series of nested subsets of \mathcal{P} , as in figure 2.

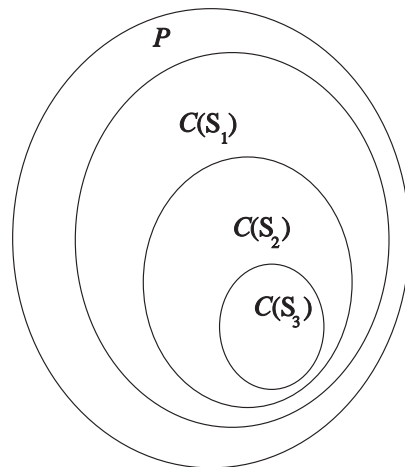


Figure 2

A nested sequence of subsets $\mathcal{C}(\mathcal{S}_1) \supset \mathcal{C}(\mathcal{S}_2) \supset \mathcal{C}(\mathcal{S}_3) \dots$ of \mathcal{P} corresponding to increasingly large sets of features $\mathcal{S}_1 \subset \mathcal{S}_2 \subset \mathcal{S}_3 \dots$

4.2 Basic Feature Selection

The basic incremental growth procedure may be outlined as follows. Every stage of the process is characterized by a set of active features \mathcal{S} . These determine a space of models

$$\mathcal{C}(\mathcal{S}) \equiv \{p \in \mathcal{P} \mid p(f) = \hat{p}(f) \text{ for all } f \in \mathcal{S}\} \quad (19)$$

The optimal model in this space, denoted by $p_{\mathcal{S}}$, is the model with the greatest entropy:

$$p_{\mathcal{S}} \equiv \operatorname{argmax}_{p \in \mathcal{C}(\mathcal{S})} H(p) \quad (20)$$

By adding feature \hat{f} to \mathcal{S} , we obtain a new set of active features $\mathcal{S} \cup \hat{f}$. Following (19), this set of features determines a set of models

$$\mathcal{C}(\mathcal{S} \cup \hat{f}) \equiv \{p \in \mathcal{P} \mid p(f) = \hat{p}(f) \text{ for all } f \in \mathcal{S} \cup \hat{f}\} \quad (21)$$

The optimal model in this space of models is

$$p_{\mathcal{S} \cup f} \equiv \operatorname{argmax}_{p \in \mathcal{C}(\mathcal{S} \cup f)} H(p) \quad (22)$$

Adding the feature \hat{f} allows the model $p_{\mathcal{S} \cup \hat{f}}$ to better account for the training sample; this results in a *gain* $\Delta L(\mathcal{S}, \hat{f})$ in the log-likelihood of the training data

$$\Delta L(\mathcal{S}, \hat{f}) \equiv L(p_{\mathcal{S} \cup \hat{f}}) - L(p_{\mathcal{S}}) \quad (23)$$

At each stage of the model-construction process, our goal is to select the candidate feature $\hat{f} \in \mathcal{F}$ which maximizes the gain $\Delta L(\mathcal{S}, \hat{f})$; that is, we select the candidate feature which, when adjoined to the set of active features \mathcal{S} , produces the greatest increase in likelihood of the training sample. This strategy is implemented in

Algorithm 2: Basic Feature Selection

Input: Collection \mathcal{F} of candidate features; empirical distribution $\tilde{p}(x, y)$

Output: Set \mathcal{S} of active features; model $p_{\mathcal{S}}$ incorporating these features

1. Start with $\mathcal{S} = \emptyset$; thus $p_{\mathcal{S}}$ is uniform
2. Do for each candidate feature $f \in \mathcal{F}$:
 - Compute the model $p_{\mathcal{S} \cup f}$ using Algorithm 1
 - Compute the gain in the log-likelihood from adding this feature using (23)
3. Check the termination condition
4. Select the feature \hat{f} with maximal gain $\Delta L(\mathcal{S}, \hat{f})$
5. Adjoin \hat{f} to \mathcal{S}
6. Compute $p_{\mathcal{S}}$ using Algorithm 1
7. Go to step 2

One issue left unaddressed by this algorithm is the termination condition. Obviously, we would like a condition which applies exactly when all the “useful” features have been selected. One reasonable stopping criterion is to subject each proposed feature to cross-validation on a held-out sample of data. If the feature does not lead to an increase in likelihood of the held-out sample of data, the feature is discarded. We will have more to say about the stopping criterion in Section 5.3.

4.3 Approximate Gains

Algorithm 2 is not a practical method for incremental feature selection. For each candidate feature $f \in \mathcal{F}$ considered in step 2, we must compute the maximum entropy model $p_{\mathcal{S} \cup f}$, a task that is computationally costly even with the efficient iterative scaling algorithm introduced earlier. We therefore introduce a modification to the algorithm, making it greedy but much more feasible. We replace the computation of the gain $\Delta L(\mathcal{S}, f)$ of a feature f with an approximation, which we will denote by $\sim \Delta L(\mathcal{S}, f)$.

Recall that a model $p_{\mathcal{S}}$ has a set of parameters λ , one for each feature in \mathcal{S} . The model $p_{\mathcal{S} \cup f}$ contains this set of parameters, plus a single new parameter α , corresponding to f .⁵

⁵ Another way to think of this is that the models $p_{\mathcal{S} \cup f}$ and $p_{\mathcal{S}}$ have the same number of parameters, but $\alpha = 0$ for $p_{\mathcal{S}}$.

Given this structure, we might hope that the optimal values for λ do not change as the feature f is adjoined to \mathcal{S} . Were this the case, imposing an additional constraint would require only optimizing the single parameter α to maximize the likelihood. Unfortunately, when a new constraint is imposed, the optimal values of *all* parameters change.

However, to make the feature-ranking computation tractable, we make the approximation that the addition of a feature f affects only α , leaving the λ -values associated with other features unchanged. That is, when determining the gain of f over the model $p_{\mathcal{S}}$, we pretend that the best model containing features $\mathcal{S} \cup f$ has the form

$$p_{\mathcal{S},f}^{\alpha} = \frac{1}{Z_{\alpha}(x)} p_{\mathcal{S}}(y|x) e^{\alpha f(x,y)}, \quad \text{for some real valued } \alpha \quad (24)$$

$$\text{where } Z_{\alpha}(x) = \sum_y p_{\mathcal{S}}(y|x) e^{\alpha f(x,y)} \quad (25)$$

The only parameter which distinguishes models of the form (24) is α . Among these models, we are interested in the one which maximizes the *approximate gain*

$$\begin{aligned} G_{\mathcal{S},f}(\alpha) &\equiv L(p_{\mathcal{S},f}^{\alpha}) - L(p_{\mathcal{S}}) \\ &= - \sum_x \tilde{p}(x) \log Z_{\alpha}(x) + \alpha \tilde{p}(f) \end{aligned} \quad (26)$$

We will denote the gain of this model by

$$\sim \Delta L(\mathcal{S}, f) \equiv \max_{\alpha} G_{\mathcal{S},f}(\alpha) \quad (27)$$

and the optimal model by

$$\sim p_{\mathcal{S} \cup f} \equiv \operatorname{argmax}_{p_{\mathcal{S},f}^{\alpha}} G_{\mathcal{S},f}(\alpha) \quad (28)$$

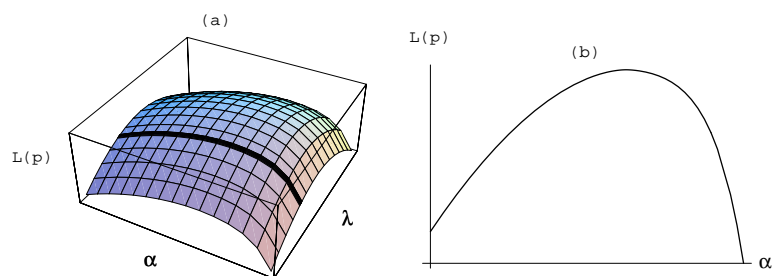
Despite the rather unwieldy notation, the idea is simple. Computing the approximate gain in likelihood from adding feature f to $p_{\mathcal{S}}$ has been reduced to a simple one-dimensional optimization problem over the single parameter α , which can be solved by any popular line-search technique such as Newton's method. This yields a great savings in computational complexity over computing the exact gain, an n -dimensional optimization problem requiring more sophisticated methods such as conjugate gradient. But the savings comes at a price: for any particular feature f , we are probably underestimating its gain, and there is a reasonable chance that we will select a feature f whose approximate gain $\sim \Delta L(\mathcal{S}, f)$ was highest and pass over the feature \hat{f} with maximal gain $\Delta L(\mathcal{S}, \hat{f})$.

A graphical representation of this approximation is provided in figure 3. Here the log-likelihood is represented as an arbitrary convex function over two parameters: λ corresponds to the "old" parameter, and α to the "new" parameter. Holding λ fixed and adjusting α to maximize the log-likelihood involves a search over the darkened line, rather than a search over the entire space of (λ, α) .

The actual algorithms, along with the appropriate mathematical framework, are presented in the appendix.

5. Case Studies

In the next few pages we discuss several applications of maximum entropy modeling within *Candide*, a fully automatic French-to-English machine translation system under

**Figure 3**

The likelihood $L(p)$ is a convex function of its parameters. If we start from a one-constraint model whose optimal parameter value is $\lambda = \lambda_0$ and consider the increase in $L_{\hat{p}}(p)$ from adjoining a second constraint with the parameter α , the exact answer requires a search over (λ, α) . We can simplify this task by holding $\lambda = \lambda_0$ constant and performing a line search over the possible values of the new parameter α . In (a), the darkened line represents the search space we restrict attention to. In (b) we show the reduced problem: a line search over α .

development at IBM. Over the past few years, we have used Candide as a test bed for exploring the efficacy of various techniques in modeling problems arising in machine translation.

We begin in Section 5.1 with a review of the general theory of statistical translation, describing in some detail the models employed in Candide. In Section 5.2 we describe how we have applied maximum entropy modeling to predict the French translation of an English word in context. In Section 5.3 we describe maximum entropy models that predict differences between French word order and English word order. In Section 5.4 we describe a maximum entropy model that predicts how to divide a French sentence into short segments that can be translated sequentially.

5.1 Review of Statistical Translation

When presented with a French sentence F , Candide's task is to find the English sentence \hat{E} which is most likely given F :

$$\hat{E} = \operatorname{argmax}_E p(E|F) \quad (29)$$

By Bayes' theorem, this is equivalent to finding \hat{E} such that

$$\hat{E} = \operatorname{argmax}_E p(F|E)p(E) \quad (30)$$

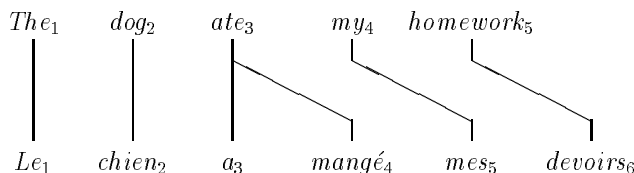
Candide estimates $p(E)$ —the probability that a string E of English words is a well-formed English sentence—using a parametric model of the English language, commonly referred to as a *language model*. The system estimates $p(F|E)$ —the probability that a French sentence F is a translation of E —using a parametric model of the process of English-to-French translation known as a *translation model*. These two models, plus a search strategy for finding the \hat{E} which maximizes (30) for some F , comprise the engine of the translation system.

We now briefly describe the translation model for the probability $P(F|E)$; a more thorough account is provided in (Brown *et al* 1991). We imagine that an English sentence E generates a French sentence F in two steps. First each word in E independently generates zero or more French words. These words are then ordered to give a French sentence F . We denote the i th word of E by e_i and the j th word of F by y_j . (We employ y_j rather than the more intuitive f_j to avoid confusion with the feature function notation.) We denote the number of words in the sentence E by $|E|$ and the number of words in the sentence F by $|F|$. The generative process yields not only the French sentence F but also an association of the words of F with the words of E . We call this association an *alignment*, and denote it by A . An alignment A is parametrized by a sequence of $|F|$ numbers a_j , with $1 \leq a_j \leq |E|$. For every word position j in F , a_j is the word position in E of the English word that generates y_j . Figure 4 depicts a typical alignment.

The probability $p(F|E)$ that F is the translation of E is expressed as the sum over all possible alignments A between E and F of the probability of F and A given E :

$$p(F|E) = \sum_A p(F, A|E) \quad (31)$$

The sum in equation (31) is computationally unwieldy; it involves a sum over all $|E|^{|F|}$ possible alignments between the words in the two sentences. For this reason we sometimes

**Figure 4**

Alignment of a French–English sentence pair. The subscripts give the position of each word in its sentence. Here $a_1 = 1$, $a_2 = 2$, $a_3 = a_4 = 3$, $a_5 = 4$, and $a_6 = 5$.

make the simplifying assumption that there exists one extremely probable alignment \hat{A} , called the “Viterbi alignment,” for which

$$p(F|E) \approx p(F, \hat{A}|E) \quad (32)$$

Given some alignment A (Viterbi or otherwise) between E and F , the probability $p(F, A|E)$ is given by

$$p(F, A|E) = \prod_{i=1}^{|E|} p(n(e_i)|e_i) \cdot \prod_{j=1}^{|F|} p(y_j|e_{a_j}) \cdot d(A|E, F) \quad (33)$$

where $n(e_i)$ denotes the number of French words aligned with e_i . In this expression

- $p(n|e)$ is the probability that the English word e generates n French words,
- $p(y|e)$ is the probability that the English word e generates the French word y ; and
- $d(A|E, F)$ is the probability of the particular order of French words.

We call the model described by equations (31) and (33) the *basic translation model*.

We take the probabilities $p(n|e)$ and $p(y|e)$ as the fundamental parameters of the model, and parametrize the distortion probability in terms of simpler distributions. (Brown *et al* 1991) describe a method of estimating these parameters to maximize the likelihood of a large bilingual corpus of English and French sentences. Their method is based on the *Estimation-Maximization* (EM) algorithm, a well-known iterative technique for maximum likelihood training of a model involving hidden statistics. For the basic translation model, the hidden information is the alignment A between E and F .

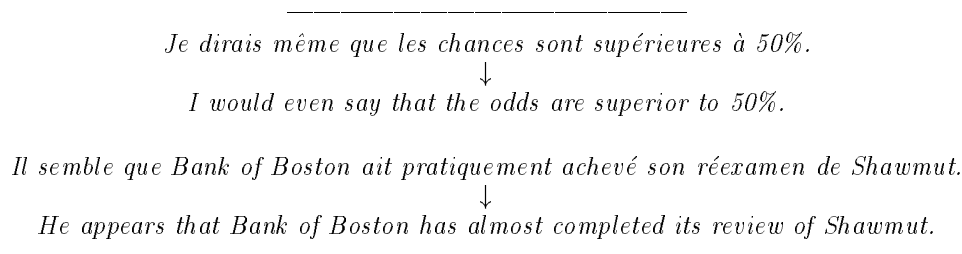
We employed the EM algorithm to estimate the parameters of the basic translation model so as to maximize the likelihood of a bilingual corpus obtained from the proceedings of the Canadian parliament. For historical reasons, these proceedings are sometimes called “Hansards.” Our Hansard corpus contains 3.6 million English-French sentence pairs for a total of a little under 100 million words in each language. Table 2 shows our parameter estimates for the translation probabilities $p(y|in)$. The basic translation model has worked admirably: given only the bilingual corpus, with no additional knowledge of the languages or any relation between them, it has uncovered some highly plausible translations.

Nevertheless, the basic translation model has one major shortcoming: it does not take the English context into account. That is, the model does not account for surrounding English words when predicting the appropriate French rendering of an English word. As we pointed out in Section 3, this is not how successful translation works. The best French translation of *in* is a function of the surrounding English words: if *a month’s time* are

Translation	Probability
<i>dans</i>	0.3004
<i>à</i>	0.2275
<i>de</i>	0.1428
<i>en</i>	0.1361
<i>pour</i>	0.0349
(OTHER)	0.0290
<i>au cours de</i>	0.0233
,	0.0154
<i>sur</i>	0.0123
<i>par</i>	0.0101
<i>pendant</i>	0.0044

Table 2

Most frequent French translations of *in* as estimated using EM-training. (OTHER) represents a catch-all classifier for any French phrase not listed, none of which had a probability exceeding 0.0043.

**Figure 5**

Typical errors encountered in using EM-based model of Brown *et. al.* in a French-to-English translation system

the subsequent words, *pendant* might be more likely, but if *the fiscal year 1992* are what follows, then *dans* is more likely. The basic model is blind to context, always assigning a probability of 0.3004 to *dans* and 0.0044 to *pendant*.

This can yield errors when Candide is called upon to translate a French sentence. Examples of two such errors are shown in Figure 5. In the first example, the system has chosen an English sentence in which the French word *supérieures* has been rendered as *superior* when *greater* or *higher* is a preferable translation. With no knowledge of context, an expert translator is also quite likely to select *superior* as the English word which generates *supérieures*. But if the expert were privy to the fact that *50%* was among the next few words, he might be more inclined to select *greater* or *higher*. Similarly, in the second example, the incorrect rendering of *Il* as *He* might have been avoided had the translation model used the fact that the word following *it* is *appears*.

5.2 Context-Dependent Word Models

In the hope of rectifying these errors, we consider the problem of context-sensitive modeling of word translation. We envision, in practice, a separate maximum entropy model, $p_e(y|x)$, for each English word e , where $p_e(y|x)$ represents the probability that an expert translator would choose y as the French rendering of e , given the surrounding English context x . This is just a slightly recast version of a longstanding problem in computational linguistics, namely sense disambiguation—the determination of a word’s sense from its context.

We begin with a training sample of English-French sentence pairs (E, F) randomly extracted from the Hansard corpus, such that E contains the English word *in*. For each sentence pair, we use the basic translation model to compute the Viterbi alignment \hat{A} between E and F . Using this alignment, we then construct an (x, y) training event. The event consists of a context x containing the six words in E surrounding *in* and a future y equal to the French word which is (according to the Viterbi alignment \hat{A}) aligned with *in*. A few actual examples of such events for *in* are depicted in Table 3.

Next we define the set of candidate features. For this application, we employ features that are indicator functions of simply described sets. Specifically, we consider functions $f(x, y)$ which are one if y is some particular French word and the context x contains a given English word, and are zero otherwise. We employ the following notation to represent these features:

translation	e_{-3}	e_{-2}	e_{-1}	e_{+1}	e_{+2}	e_{+3}
<i>dans</i>	<i>the</i>	<i>committee</i>	<i>stated</i>	<i>· a</i>	<i>letter</i>	<i>to</i>
<i>à</i>	<i>work</i>	<i>was</i>	<i>required</i>	<i>· respect</i>	<i>of</i>	<i>the</i>
<i>au cours de</i>				<i>· the</i>	<i>fiscal</i>	<i>year</i>
<i>dans</i>	<i>by</i>	<i>the</i>	<i>government</i>	<i>· the</i>	<i>same</i>	<i>postal</i>
<i>à</i>	<i>of</i>	<i>diphtheria</i>	<i>reported</i>	<i>· Canada</i>	<i>,</i>	<i>by</i>
<i>de</i>	<i>not</i>	<i>given</i>	<i>notice</i>	<i>· the</i>	<i>ordinary</i>	<i>way</i>

Table 3

Several actual training events for the maximum entropy translation model for *in*, extracted from the transcribed proceedings of the Canadian parliament.

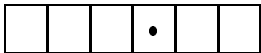
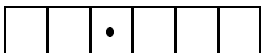
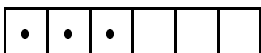
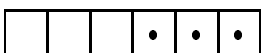
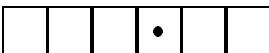

Template	Number of actual features	$f(x, y) = 1$ if and only if ...
1	$ \mathcal{V}_{\mathcal{F}} $	$y = \diamond$
2	$ \mathcal{V}_{\mathcal{F}} \cdot \mathcal{V}_{\mathcal{E}} $	$y = \diamond$ and $\square \in$ 
3	$ \mathcal{V}_{\mathcal{F}} \cdot \mathcal{V}_{\mathcal{E}} $	$y = \diamond$ and $\square \in$ 
4	$ \mathcal{V}_{\mathcal{F}} \cdot \mathcal{V}_{\mathcal{E}} $	$y = \diamond$ and $\square \in$ 
5	$ \mathcal{V}_{\mathcal{F}} \cdot \mathcal{V}_{\mathcal{E}} $	$y = \diamond$ and $\square \in$ 

Table 4
 Feature templates for word-translation modeling. $|\mathcal{V}_{\mathcal{E}}|$ is the size of the English vocabulary; $|\mathcal{V}_{\mathcal{F}}|$ the size of the French vocabulary.

$$f_1(x, y) = \begin{cases} 1 & y = en \text{ and } April \in \text{  } \\ 0 & \text{otherwise} \end{cases}$$

$$f_2(x, y) = \begin{cases} 1 & y = pendant \text{ and } weeks \in \text{  } \\ 0 & \text{otherwise} \end{cases}$$

Here $f_1 = 1$ when *April* follows *in* and *en* is the translation of *in*; $f_2 = 1$ when *weeks* is one of the three words following *in* and *pendant* is the translation.

The set of features under consideration is vast, but may be expressed in abbreviated form in Table 4. In the table, the symbol \diamond is a placeholder for a possible French word and the symbol \square is placeholder for a possible English word. The feature f_1 mentioned above is thus derived from template 2 with $\diamond = en$ and $\square = April$; the feature f_2 is derived from template 5 with $\diamond = pendant$ and $\square = weeks$. If there are $|\mathcal{V}_{\mathcal{E}}|$ total English words and $|\mathcal{V}_{\mathcal{F}}|$ total French words, there are $|\mathcal{V}_{\mathcal{F}}|$ template-1 features, and $|\mathcal{V}_{\mathcal{E}}| \cdot |\mathcal{V}_{\mathcal{F}}|$ features of templates 2,3,4 and 5.

Template 1 features give rise to constraints that enforce equality between the probability of any French translation y of *in* according to the model and the probability of that translation in the empirical distribution. Examples of such constraints are

$$\begin{aligned} p(y = dans) &= \tilde{p}(y = dans) \\ p(y = \grave{a}) &= \tilde{p}(y = \grave{a}) \\ p(y = de) &= \tilde{p}(y = de) \\ p(y = en) &= \tilde{p}(y = en) \\ &\vdots \end{aligned}$$

A maximum entropy model that uses only template 1 features predicts each French

translation y with the probability $\tilde{p}(y)$ determined by the empirical data. This is exactly the distribution employed by the basic translation model.

Since template 1 features are independent of x , the maximum entropy model which employs only constraints derived from template 1 features takes no account of contextual information in assigning a probability to y . When we include constraints derived from template 2 features, we take our first step towards a context-dependent model. Rather than simply constraining the expected probability of a French word y to equal its empirical probability, these constraints require that the expected *joint* probability of the English word immediately following *in* and the French rendering of *in* be equal to its empirical probability. An example of a template 2 constraint is

$$p(y = \textit{pendant}, e_{+1} = \textit{several}) = \tilde{p}(y = \textit{pendant}, e_{+1} = \textit{several})$$

A maximum entropy model that incorporates this constraint will predict the translations of *in* in a manner consistent with whether or not the following word is *several*. In particular, if in the empirical sample, the presence of *several* led to a greater probability for *pendant*, this will be reflected in a maximum entropy model incorporating this constraint. We have thus taken our first step toward context-sensitive translation modeling.

Templates 3, 4 and 5 consider, each in a different way, various parts of the context. For instance, template 5 constraints allow us to model how an expert translator is biased by the appearance of a word *somewhere* in the three words following the word he is translating. If *house* appears within the next three words (e.g. the phrases *in the house* and *in the red house*), then *dans* might be a more likely translation. On the other hand, if *year* appears within the same window of words (as in *in the year 1941* or *in that fateful year*), then *au cours de* might be more likely. Together, the five constraint templates allow the model to condition its assignment of probabilities on a window of six words around e_0 , the word in question.

We constructed a maximum entropy model $p_{in}(y|x)$ by the iterative model-growing method described in Section 4. The automatic feature selection algorithm first selected a template 1 constraint for each of the translations of *in* seen in the sample (12 in all), thus constraining the model's expected probability of each of these translations to their empirical probabilities. The next few constraints selected by the algorithm are shown in Table 5. The first column gives the identity of the feature whose expected value is constrained; the second column gives $\sim\Delta L(\mathcal{S}, f)$, the approximate increase in the model's log-likelihood on the data as a result of imposing this constraint; the third column gives $L(p)$, the log-likelihood after adjoining the feature and recomputing the model.

Let us consider the fifth row in the table. This constraint requires that the model's expected probability of *dans*, if one of the three words to the right of *in* is the word *speech*, is equal to that in the empirical sample. Before imposing this constraint on the model during the iterative model-growing process, the log-likelihood of the current model on the empirical sample was -2.8703 bits. The feature selection algorithm described in Section 4 calculated that if this constraint were imposed on the model, the log-likelihood would rise by approximately 0.019059 bits; since this value was higher than for any other constraint considered, the constraint was selected. After applying iterative scaling to recompute the parameters of the new model, the likelihood of the empirical sample rose to -2.8525 bits, an increase of 0.0178 bits.

Table 6 lists the first few selected features for the model for translating the English word *run*. The "Hansard flavor"—the rather specific domain of parliamentary discourse related to Canadian affairs—is easy to detect in many of the features in this Table 5.

It is not hard to incorporate the maximum entropy word translation models into a translation model $p(F|E)$ for a French sentence given an English sentence. We merely

	Feature $f(x, y)$	$\sim \Delta L(\mathcal{S}, f)$	$L(p)$						
$y=\grave{a}$ and <i>Canada</i> \in	<table border="1"><tr><td></td><td></td><td></td><td>•</td><td></td><td></td></tr></table>				•			0.0415	-2.9674
			•						
$y=\grave{a}$ and <i>House</i> \in	<table border="1"><tr><td></td><td></td><td></td><td>•</td><td></td><td></td></tr></table>				•			0.0361	-2.9281
			•						
$y=en$ and <i>the</i> \in	<table border="1"><tr><td></td><td></td><td></td><td>•</td><td></td><td></td></tr></table>				•			0.0221	-2.8944
			•						
$y=pour$ and <i>order</i> \in	<table border="1"><tr><td></td><td></td><td></td><td>•</td><td></td><td></td></tr></table>				•			0.0224	-2.8703
			•						
$y=dans$ and <i>speech</i> \in	<table border="1"><tr><td></td><td></td><td></td><td>•</td><td>•</td><td>•</td></tr></table>				•	•	•	0.0190	-2.8525
			•	•	•				
$y=dans$ and <i>area</i> \in	<table border="1"><tr><td></td><td></td><td></td><td>•</td><td>•</td><td>•</td></tr></table>				•	•	•	0.0153	-2.8377
			•	•	•				
$y=de$ and <i>increase</i> \in	<table border="1"><tr><td>•</td><td>•</td><td>•</td><td></td><td></td><td></td></tr></table>	•	•	•				0.0151	-2.8209
•	•	•							
$y=[verb\ marker]$ and <i>my</i> \in	<table border="1"><tr><td></td><td></td><td></td><td>•</td><td></td><td></td></tr></table>				•			0.0141	-2.8034
			•						
$y=dans$ and <i>case</i> \in	<table border="1"><tr><td></td><td></td><td></td><td>•</td><td>•</td><td>•</td></tr></table>				•	•	•	0.0116	-2.7918
			•	•	•				
$y=au\ cours\ de$ and <i>year</i> \in	<table border="1"><tr><td></td><td></td><td></td><td>•</td><td>•</td><td>•</td></tr></table>				•	•	•	0.0104	-2.7792
			•	•	•				

Table 5 Maximum entropy model to predict French translation of *in*. Features shown here were the first non template 1 features selected. *[verb marker]* denotes a morphological marker inserted to indicate the presence of a verb as the next word.

	Feature $f(x, y)$	$\sim \Delta L(\mathcal{S}, f)$	$L(p)$						
$y = \text{épuiser}$ and $out \in$	<table border="1"><tr><td></td><td></td><td></td><td>•</td><td>•</td><td>•</td></tr></table>				•	•	•	0.0252	-4.8499
			•	•	•				
$y = \text{manquer}$ and $out \in$	<table border="1"><tr><td></td><td></td><td></td><td>•</td><td>•</td><td>•</td></tr></table>				•	•	•	0.0221	-4.8201
			•	•	•				
$y = \text{écouler}$ and $time \in$	<table border="1"><tr><td></td><td></td><td></td><td>•</td><td>•</td><td>•</td></tr></table>				•	•	•	0.0157	-4.7969
			•	•	•				
$y = \text{accumuler}$ and $up \in$	<table border="1"><tr><td></td><td></td><td></td><td>•</td><td></td><td></td></tr></table>				•			0.0149	-4.7771
			•						
$y = \text{nous}$ and $we \in$	<table border="1"><tr><td></td><td></td><td>•</td><td></td><td></td><td></td></tr></table>			•				0.0140	-4.7582
		•							
$y = \text{aller}$ and $counter \in$	<table border="1"><tr><td></td><td></td><td></td><td>•</td><td>•</td><td>•</td></tr></table>				•	•	•	0.0131	-4.7445
			•	•	•				
$y = \text{candidat}$ and $for \in$	<table border="1"><tr><td></td><td></td><td></td><td>•</td><td>•</td><td>•</td></tr></table>				•	•	•	0.0124	-4.7295
			•	•	•				
$y = \text{diriger}$ and $the \in$	<table border="1"><tr><td></td><td></td><td></td><td>•</td><td>•</td><td>•</td></tr></table>				•	•	•	0.0123	-4.7146
			•	•	•				

Table 6
Maximum entropy model to predict French translation of *to run*: top-ranked features not from template 1

Je dirais même que les chances sont supérieures à 50%.

↓

I would even say that the odds are greater than 50%.

Il semble que Bank of Boston ait pratiquement achevé son réexamen de Shawmut.

↓

It appears that Bank of Boston has almost completed its review of Shawmut .

Figure 6

Improved French-to-English translations resulting from maximum entropy-based system

replace the simple context-independent models $p(y|e)$ used in the basic translation model (33) with the more general context-dependent models $p_e(y|x)$:

$$p(F, A|E) = \prod_{i=1}^{|E|} p(n(e_i)|e_i) \cdot \prod_{j=1}^{|F|} p_{e_{a_j}}(y_j|x_{a_j}) \cdot d(A|E, F)$$

where x_{a_j} denotes the context of the English word e_{a_j} .

Figure 6 illustrates how using this improved translation model in the Candide system led to improved translations for the two sample sentences given earlier.

5.3 Segmentation

Though an ideal machine translation system could devour input sentences of unrestricted length, a typical stochastic system must cut the French sentence into polite lengths before digesting them. If the processing time is exponential in the length of the input passage (as is the case with the Candide system), then not splitting the French sentence into reasonably-sized segments would result in an exponential slowdown in translation.

Thus, a common task in machine translation is to find safe positions at which to split input sentences in order to speed the translation process. “Safe” is a vague term; one might, for instance, reasonably define a safe segmentation as one which results in coherent blocks of words. For our purposes, however, a safe segmentation is dependent on the Viterbi alignment \hat{A} between the input French sentence F and its English translation E .

We define a *rift* as a position j in F such that for all $k < j$, $a_k \leq a_j$ and for all $k > j$, $a_k \geq a_j$. In other words, the words to the left of the French word y_j are generated by words to the left of the English word e_{a_j} , and the words to the right of y_j are generated by words to the right of e_{a_j} . In the alignment of figure 4, for example, there are rifts at positions $j = 1, 2, 4, 5$ in the French sentence. One visual method of determining whether a rift occurs after the French word j is to try to trace a line from the last letter of y_j up to the last letter of e_{a_j} ; if the line can be drawn without intersecting any alignment lines, position f is a rift.

Using our definition of rifts, we can redefine a “safe” segmentation as one in which the segment boundaries are located only at rifts. Figure 7 illustrates an unsafe segmentation, in which a segment boundary (denoted by the || symbol) lies between a and *mangé*, where there is no rift. Figure 8, on the other hand, illustrates a safe segmentation.

The reader will notice that a safe segmentation does not necessarily result in semantically coherent segments: *mes* and *devoirs* are certainly part of one logical unit,

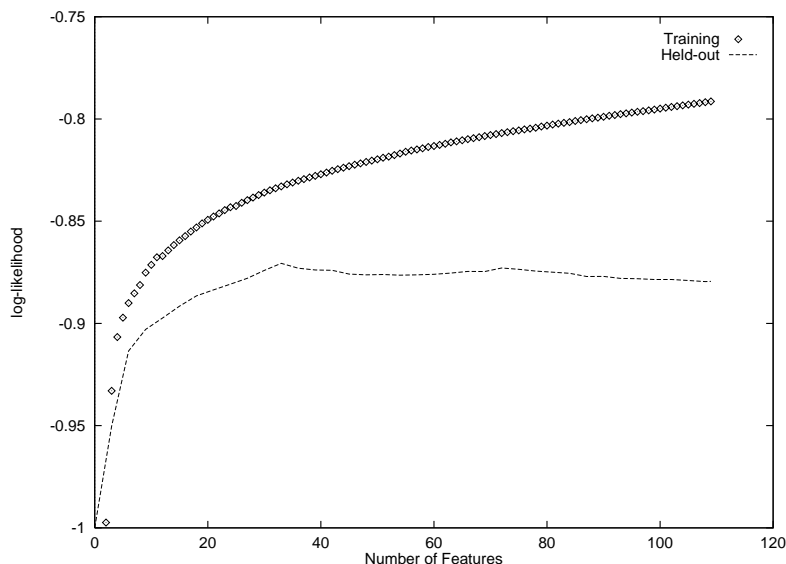


Figure 10
Change in log-likelihood during segmenting model-growing. (Overtraining begins to occur at about 40 features)

and we measure the worth of a model by the log-likelihood $L_{\tilde{p}}(p)$. During the iterative model-growing procedure, the algorithm selects constraints on the basis of how much they increase this objective function. As the algorithm proceeds, more and more constraints are imposed on the model p , bringing it into ever-stricter compliance with the empirical data $\tilde{p}(x, y)$. This is useful to a point; insofar as the empirical data embodies the expert knowledge of the French segmenter, we would like to incorporate this knowledge into a model. But the data contains only so much expert knowledge; the algorithm should terminate when it has extracted this knowledge. Otherwise, the model $p(y|x)$ will begin to fit itself to quirks in the empirical data.

A standard approach in statistical modeling to avoid the problem of overfitting the training data is employ cross-validation techniques. Separate the training data $\tilde{p}(x, y)$ into a training portion, \tilde{p}_r , and a heldout portion, \tilde{p}_h . Use only \tilde{p}_r in the model-growing process; that is, select features based on how much they increase the likelihood $L_{\tilde{p}_r}(p)$. As the algorithm progresses, $L_{\tilde{p}_r}(p)$ thus increases monotonically. As long as each new constraint imposed allows p to better account for the random process which generated both \tilde{p}_r and \tilde{p}_h , the quantity $L_{\tilde{p}_h}(p)$ also increases. At the point when overfitting begins, however, the new constraints no longer help p model the random process, but instead require p to model the noise in the sample \tilde{p}_r itself. At this point, $L_{\tilde{p}_r}(p)$ continues to rise, but $L_{\tilde{p}_h}(p)$ no longer does. It is at this point that the algorithm should terminate.

Figure 10 illustrates the change in log-likelihood of training data $L_{\tilde{p}_r}(p)$ and held-out data $L_{\tilde{p}_h}(p)$. Had the algorithm terminated when the log-likelihood of the held-out data stopped increasing, the final model p would contain slightly less than 40 features.

We have employed this segmenting model as a component in a French-English machine translation system in the following manner. The model assigns to each position in the French sentence a score, $p(\mathbf{r}\mathbf{i}\mathbf{f}\mathbf{t} | x)$, which is a measure of how appropriate a split would be at that location. A dynamic programming algorithm then selects, given the “appropriateness” score at each position and the requirement that no segment may contain more than 10 words, an optimal (or, at least, reasonable) splitting of the sentence.

Monsieur l'Orateur

j'aimerais poser une question au
Ministre des Transports.

— —

A quelle date le
nouveau règlement devrait il entrer en vigueur?

— —

Quels furent les critères utilisés
pour l'évaluation
de ces biens.

— —

Nous
savons que si nous pouvions contrôler la folle avoine
dans l'ouest du Canada, en
un an nous
augmenterions notre rendement en
céréales de 1 milliard de dollars.

Figure 11
Maximum entropy segmenter behavior on four sentences selected at random from the Hansard data

Figure 11 shows the system's segmentation of four sentences selected at random from the Hansard data. We remind the reader to keep in mind when evaluating Figure 11 that the segmenter's task is not to produce logically coherent blocks of words, but to divide the sentence into blocks which can be translated sequentially from left to right.

5.4 Word Reordering

Translating a French sentence into English involves not only selecting appropriate English renderings of the words in the French sentence, but also selecting an ordering for the English words. This order is often very different from the French word order. One way Candide captures word-order differences in the two languages is to allow for alignments with crossing lines. In addition, Candide performs, during a pre-processing stage, a reordering step which shuffles the words in the input French sentence into an order more closely resembling English word order.

One component of this word reordering step deals with French phrases which have the NOUN *de* NOUN form. For some NOUN *de* NOUN phrases, the best English translation is nearly word for word: *conflit d'intérêt*, for example, is almost always rendered as *conflict of interest*. For other phrases, however, the best translation is obtained by interchanging the two nouns and dropping the *de*. The French phrase *taux d'intérêt*, for example, is best rendered as *interest rate*. Table 7 gives several examples of NOUN *de* NOUN phrases together with their most appropriate English translations.

In this section we describe a maximum entropy model which, given a French NOUN *de* NOUN phrase, estimates the probability that the best English translation involves an interchange of the two nouns. We begin with a sample of English-French sentence pairs (E, F) randomly extracted from the Hansard corpus, such that F contains a *de* phrase. For each sentence pair we use the basic translation model to compute the Viterbi alignment \hat{A} between the words in E and F . Using \hat{A} we construct an (x, y) training event as follows. We let the context x be the pair of French nouns $(\text{NOUN}_L, \text{NOUN}_R)$. We let y be

word-for-word phrases	
<i>somme d'argent</i>	<i>sum of money</i>
<i>pays d'origin</i>	<i>country of origin</i>
<i>question de privilège</i>	<i>question of privilege</i>
<i>conflit d'intérêt</i>	<i>conflict of interest</i>
interchanged phrases	
<i>bureau de poste</i>	<i>post office</i>
<i>taux d'intérêt</i>	<i>interest rate</i>
<i>compagnie d'assurance</i>	<i>insurance company</i>
<i>gardien de prison</i>	<i>prison guard</i>

Table 7
NOUN *de* NOUN phrases and their English equivalents

Template	Number of actual features	$f(x, y) = 1$ if and only if ...
1	$2 \mathcal{V}_{\mathcal{F}} $	$y = \diamond$ and $\text{NOUN}_L = \square$
2	$2 \mathcal{V}_{\mathcal{F}} $	$y = \diamond$ and $\text{NOUN}_R = \square$
3	$2 \mathcal{V}_{\mathcal{F}} ^2$	$y = \diamond$ and $\text{NOUN}_L = \square_1$ and $\text{NOUN}_R = \square_2$

Table 8
Template features for NOUN *de* NOUN model

no-interchange if the English translation is a word-for-word translation of the French phrase and $y = \text{interchange}$ if the order of the nouns in the English and French phrases are interchanged.

We define candidate features based upon the *template features* shown in Table 8. In this table, the symbol \diamond is a placeholder for either **interchange** or **no-interchange** and the symbols \square_1 and \square_2 are placeholders for possible French words. If there are $|\mathcal{V}_{\mathcal{F}}|$ total French words, there are $2|\mathcal{V}_{\mathcal{F}}|$ possible features of templates 1 and 2 and $2|\mathcal{V}_{\mathcal{F}}|^2$ features of template 3.

Template 1 features consider only the left noun. We expect these features to be relevant when the decision of whether to interchange the nouns is influenced by the identity of the left noun. For example, including the template 1 feature

$$f(x, y) = \begin{cases} 1 & y = \text{interchange} \text{ and } \text{NOUN}_L = \textit{systeme} \\ 0 & \text{otherwise} \end{cases}$$

gives the model sensitivity to the fact that the nouns in French NOUN *de* NOUN phrases which begin with *systeme* (such as *systeme de surveillance* and *systeme de quota*) are more likely to be interchanged in the English translation. Similarly, including the template 1 feature

$$f(x, y) = \begin{cases} 1 & y = \text{no-interchange} \text{ and } \text{NOUN}_L = \textit{mois} \\ 0 & \text{otherwise} \end{cases}$$

gives the model sensitivity to the fact that French NOUN *de* NOUN phrases which begin with *mois*, such as *mois de mai* (month of May) are more likely to be translated word for word.

Template 3 features are useful in dealing with translating NOUN *de* NOUN phrases in which the interchange decision is influenced by both nouns. For example, NOUN *de* NOUN phrases ending in *intéret* are sometimes translated word for word, as in *conflit d'intéret* (*conflict of interest*) and are sometimes interchanged, as in *taux d'intéret* (*interest rate*).

We used the feature-selection algorithm of section 4 to construct a maximum entropy model from candidate features derived from templates 1,2 and 3. The model was grown on 10,000 training events randomly selected from the Hansard corpus. The final model contained 358 constraints.

To test the model, we constructed a NOUN *de* NOUN word-reordering module which interchanges the order of the nouns if $p(\text{interchange} | x) > 0.5$ and keeps the order the same otherwise. Table 9 compares performance on a suite of test data against a baseline NOUN *de* NOUN reordering module which never the swaps the word order.

Table 12 shows some randomly-chosen NOUN *de* NOUN phrases extracted from this test suite along with $p(\text{interchange}|x)$, the probability which the model assigned to inversion. On the right are phrases such as *saison d'hiver* for which the model strongly

Test data	Simple Model Accuracy	Maximum Entropy Model Accuracy
50,229 not interchanged	100%	93.5%
21,326 interchanged	0%	49.2%
71,555 total	70.2%	80.4%

Table 9
 NOUN *de* NOUN model performance: simple approach vs. maximum entropy

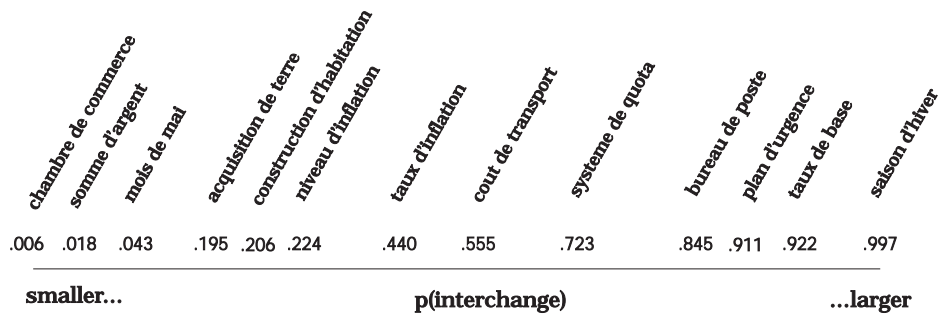


Figure 12
 Predictions of the NOUN *de* NOUN interchange model on phrases selected from a corpus unseen during the training process

predicted an inversion. On the left are phrases which the model strongly prefers not to interchange, such as *somme d'argent*, *abus de privilège* and *chambre de commerce*. Perhaps most intriguing are those phrases which lie in the middle, such as *taux d'inflation*, which can translate either to *inflation rate* or *rate of inflation*.

6. Conclusion

We began by introducing the building blocks of maximum entropy modeling—real-valued features and constraints built from these features. We then discussed the maximum entropy principle. This principle instructs us to choose, among all the models consistent with the constraints, the model with the greatest entropy. We observed that this model was a member of an exponential family with one adjustable parameter for each constraint. The optimal values of these parameters are obtained by maximizing the likelihood of the training data. Thus two different philosophical approaches—maximum entropy and maximum likelihood—yield the same result: the model with the greatest entropy consistent with the constraints is the *same* as the exponential model which best predicts the sample of data.

We next discussed algorithms for constructing maximum entropy models, concentrating our attention on the two main problems facing would-be modelers: selecting a set of features to include in a model, and computing the parameters of a model which contains these features. The general feature-selection is too slow in practice, and we presented several techniques for making the algorithm feasible.

In the second part of this paper we described several applications of our algorithms, concerning modeling tasks arising in *Candide*, an automatic machine-translation system under development at IBM. These applications demonstrate the efficacy of maximum entropy techniques for performing context-sensitive modeling.

Acknowledgments

The authors wish to thank Harry Printz and John Lafferty for suggestions and comments on a preliminary draft of this paper, and Jerome Bellegarda for providing expert French knowledge.

References

- Bahl, L., Brown, P., de Souza, P., Mercer, R. (1989) A tree-based statistical language model for natural language speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 7.
- Berger, A., Brown, P., Della Pietra, S., Della Pietra, V., Gillett, J., Lafferty, J., Printz, H., Ureš, L. (1994) The Candide system for machine translation. *Proceedings of the ARPA Conference on Human Language Technology*, Plainsborough, New Jersey.
- Black, E., Jelinek, F., Lafferty, J., Magerman, D., Mercer, R. and Roukos, S. (1992) Towards history-based grammars: using richer models for probabilistic parsing. *Proceedings of the DARPA Speech and Natural Language Workshop*, Arden House, New York.
- Brown, D. (1959) A note on approximations to discrete probability distributions. *Information and Control*, vol. 2, 386–392.
- Brown, P., Della Pietra, S., Della Pietra, V. Mercer, R. (1993) The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, vol. 19, no. 2, 263–311.
- Brown, P., Cocke, J. Della Pietra, S., Della Pietra, V., Jelinek, F., Lafferty, J., Mercer, R., and Roossin, P. (1990) A statistical approach to machine translation. *Computational Linguistics*, vol. 16, 79–85.
- Brown, P., Della Pietra, V, de Souza, P., Mercer, R. (1990) Class-based n-gram models of natural language. *Proceedings of the IBM Natural Language ITL*, 283–298.

- Brown, P., Della Pietra, S., Della Pietra, V., Mercer, R. (1991) A statistical approach to sense disambiguation in machine translation. *DARPA Workshop on Speech and Natural Language*, 146–151.
- Cover, T. and Thomas, J. (1991) *Elements of Information Theory*. John Wiley & Sons.
- Csiszár, I. (1975) I-Divergence geometry of probability distributions and minimization problems, *The Annals of Probability*, vol. 3, No. 1, 146–158.
- ibid.* (1989) A geometric interpretation of Darroch and Ratcliff’s generalized iterative scaling. *The Annals of Statistics*, vol. 17, No. 3, 1409–1413.
- Csiszár, L. and Tusnády, G. (1984) Information geometry and alternating minimization procedures. *Statistics & Decisions, Supplemental Issue*, no. 1, 205–237.
- Darroch, J.N. and Ratcliff, D. (1972) Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, no. 43, 1470–1480.
- Della Pietra, S., Della Pietra, V., Gillett, J., Lafferty, J., Printz, H., Ureš, L. (1994) Inference and estimation of a long-range trigram model. *Second International Symposium on Grammatical Inference*, Alicante, Spain.
- Della Pietra, S., Della Pietra, V., Lafferty, J. Inducing features of random fields, (1995) CMU Technical Report CMU-CS-95-144.
- Dempster, A.P., Laird, N.M., and Rubin, D.B. (1977) Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, vol 39, no. B, 1–38.
- Guiasu, S. and Shenitzer, A. (1985) The principle of maximum entropy. *The Mathematical Intelligencer*, vol. 7, no. 1.
- Jaynes, E.T. (1990) Notes on present status and future prospects. In Grandy, W.T., and Schick, L.H. *Maximum Entropy and Bayesian Methods*. Kluwer. 1–13.
- Jelinek, F. and Mercer, R. L. (1980) Interpolated estimation of Markov source parameters from sparse data. In *Proceedings, Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands.
- Lucassen, J. and Mercer, R. (1984) An information theoretic approach to automatic determination of phonemic baseforms. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, . San Diego, CA, 42.5.1–42.5.4.
- Merialdo, B. (1990) Tagging text with a probabilistic model. *Proceedings of the IBM Natural Language ITL*, Paris, France, 161–172.
- Nádas, A. Mercer, R., Bahl, L., Bakis, R., Cohen, P., Cole, A., Jelinek, F., and Lewis, B. (1981) Continuous speech recognition with automatically selected acoustic prototypes obtained by either bootstrapping or clustering. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Atlanta, GA, 1153–1155.
- Sokolnikoff, I. S., Redheffer, R. M. (1966) *Mathematics of Physics and Modern Engineering*, Second Edition, McGraw-Hill Book Company.

Appendix: Efficient Algorithms for Feature Selection

Computing the Approximate Gain of One Feature

This section picks up where section 4 left off, describing in some detail a set of algorithms which implement the feature selection process efficiently.

We first describe an iterative algorithm for computing $\sim\Delta L(\mathcal{S}, f) \equiv \max_{\alpha} G_{\mathcal{S},f}(\alpha)$ for a candidate feature f . The algorithm is based on the fact that the maximum of $G_{\mathcal{S},f}(\alpha)$ occurs (except in rare cases) at the unique value α^* at which the derivative $G'_{\mathcal{S},f}(\alpha^*)$ is zero. To find this zero we apply Newton’s iterative root-finding method. An important twist is that we do *not* use the updates obtained by applying Newton’s method directly in the variable α . This is because there is no guarantee that $G_{\mathcal{S},f}(\alpha_n)$ increases monotonically for such updates. Instead, we use updates derived by applying Newton’s method in the variables e^{α} or $e^{-\alpha}$. A convexity argument shows that using these updates the sequence of $G_{\mathcal{S},f}(\alpha_n)$ converges monotonically to the maximum approximate gain $\sim\Delta L(\mathcal{S}, f) \equiv G_{\mathcal{S},f}(\alpha^*)$ and that α_n increases monotonically to α^* .

The value α^* that maximizes $G_{\mathcal{S},f}(\alpha)$ can be found by solving the equation $G'_{\mathcal{S},f}(\alpha^*) = 0$. Moreover, if α_n is any sequence for which $G'_{\mathcal{S},f}(\alpha_n)$ converges *monotonically* to 0, then

$G_{\mathcal{S},f}(\alpha_n)$ will increase monotonically. This is a consequence of the convexity of $G_{\mathcal{S},f}(\alpha)$ in α .

We can solve an equation $g(\alpha) = 0$ by Newton's method, which produces a sequence α_n by the recurrence given in (18), repeated here for convenience:

$$\alpha_{n+1} = \alpha_n - \frac{g(\alpha_n)}{g'(\alpha_n)} \quad (34)$$

If we start with α_0 sufficiently close to α_* , then the sequence α_n will converge to α_* and $g(\alpha_n)$ will converge to zero. In general, though, the $g(\alpha_n)$ will not be monotonic. However, it can be shown that the sequence is monotonic in the following important cases: if $\alpha_0 \leq \alpha_*$ and $g(\alpha)$ is either decreasing and convex- \cup or increasing and convex- \cap .

The function $G'_{\mathcal{S},f}(\alpha)$ is neither convex- \cap or convex- \cup as a function of α . However, it can be shown (by taking derivatives) that $G'_{\mathcal{S},f}(\alpha)$ is decreasing and convex- \cup in e^α , and is increasing and convex- \cap in $e^{-\alpha}$. Thus, if $\alpha^* > 0$ so that $e^0 < e^{\alpha^*}$, we can apply Newton's method in e^α to obtain a sequence of α_n for which $G'_{\mathcal{S},f}(\alpha_n)$ increases monotonically to zero. Similarly, if $\alpha^* < 0$ so that $e^0 < e^{-\alpha^*}$, we can apply Newton's method in $e^{-\alpha}$ to obtain a sequence α_n for which $G'_{\mathcal{S},f}(\alpha_n)$ decreases monotonically to zero. In either case, $G_{\mathcal{S},f}(\alpha_n)$ increases monotonically to its maximum $G_{\mathcal{S},f}(\alpha^*)$.

The updates resulting from Newton's method applied in the variable $e^{r\alpha}$, for $r = 1$ or $r = -1$ are easily computed:

$$\alpha_{n+1} = \alpha_n + \frac{1}{r} \log \left(1 - \frac{1}{r} \frac{G'_{\mathcal{S},f}(\alpha_n)}{G''_{\mathcal{S},f}(\alpha_n)} \right) \quad (35)$$

In order to solve the recurrence in (35), we need to compute $G'_{\mathcal{S},f}$ and $G''_{\mathcal{S},f}$. The zeroth, first and second derivatives of G are

$$G_{\mathcal{S},f}(\alpha) = - \sum_x \tilde{p}(x) \log Z_\alpha(x) + \alpha \tilde{p}(f) \quad (36)$$

$$G'_{\mathcal{S},f}(\alpha) = \tilde{p}(f) - \sum_x \tilde{p}(x) p_{\mathcal{S},f}^\alpha(f|x) \quad (37)$$

$$G''_{\mathcal{S},f}(\alpha) = - \sum_x \tilde{p}(x) p_{\mathcal{S},f}^\alpha((f - p_{\mathcal{S},f}^\alpha(f|x))^2|x) \quad (38)$$

$$\text{where } p_{\mathcal{S},f}^\alpha(h|x) \equiv \sum_y p_{\mathcal{S},f}^\alpha(y|x) h(x,y) \quad (39)$$

With these in place, we are ready to enumerate

Algorithm 3: Computing the Gain of a Single Feature

Input: Empirical distribution $\tilde{p}(x, y)$; initial model $p_{\mathcal{S}}$; candidate feature f

Output: Approximate gain $\sim \Delta L(\mathcal{S}, f)$ of feature f

1. Let

$$r = \begin{cases} 1 & \text{if } \tilde{p}(f) \leq p_{\mathcal{S}}(f) \\ -1 & \text{otherwise} \end{cases} \quad (40)$$

2. Set $\alpha_0 \leftarrow 0$

3. Repeat the following until $G_{\mathcal{S},f}(\alpha_n)$ has converged:

Compute α_{n+1} from α_n using (35)

Compute $G_{\mathcal{S},f}(\alpha_{n+1})$ using (26)

4. Set $\sim\Delta L(\mathcal{S}, f) \leftarrow G_{\mathcal{S},f}(\alpha_n)$

Computing Approximate Gains in Parallel

For the purpose of incremental model growing as outlined in Algorithm 2, we need to compute the maximum approximate gain $\sim\Delta L(\mathcal{S}, f)$ for *each* candidate feature $f \in \mathcal{F}$. One obvious approach is to cycle through all candidate features and apply Algorithm 3 for each one sequentially. Since Algorithm 3 requires one pass through every event in the training sample per iteration, this could entail millions of passes through the training sample. Because a significant cost often exists for reading the training data—if the data cannot be stored in memory but must be accessed from disk, for example—an algorithm which passes a minimal number of times through the data may be of some utility. We now give a parallel algorithm specifically tailored to this scenario.

Algorithm 4: Computing Approximate Gains for A Collection of Features

Input: Collection \mathcal{F} of candidate features; empirical distribution $\tilde{p}(x, y)$;
initial model $p_{\mathcal{S}}$

Output: Approximate gain $\sim\Delta L(\mathcal{S}, f)$ for each candidate feature $f \in \mathcal{F}$

1. For each $f \in \mathcal{F}$, calculate $\tilde{p}(f)$, the expected value of f in the training data
2. For each x , determine the set $\mathcal{F}(x) \subseteq \mathcal{F}$ of f that are *active* for x :

$$\mathcal{F}(x) \equiv \{f \in \mathcal{F} \mid f(x, y)p_{\mathcal{S}}(y|x)\tilde{p}(x) > 0 \text{ for some } y\} \quad (41)$$

3. For each f , let

$$r(f) = \begin{cases} 1 & \text{if } \tilde{p}(f) \leq p_{\mathcal{S}}(f) \\ -1 & \text{otherwise} \end{cases} \quad (42)$$

4. For each $f \in \mathcal{F}$, initialize $\alpha(f) \leftarrow 0$
5. Repeat the following until $\alpha(f)$ converges for each $f \in \mathcal{F}$:
 - (a) For each $f \in \mathcal{F}$, set

$$\begin{aligned} G'(f) &\leftarrow \tilde{p}(f) \\ G''(f) &\leftarrow 0 \end{aligned}$$

- (b) For each x , do the following:

For each $f \in \mathcal{F}(x)$, update $G'(f)$ and $G''(f)$ by

$$G'(f) \leftarrow G'(f) - \tilde{p}(x)p_{\mathcal{S},f}^{\alpha}(f|x) \quad (43)$$

$$G''(f) \leftarrow G''(f) - \tilde{p}(x)p_{\mathcal{S},f}^{\alpha}((f - p_{\mathcal{S},f}^{\alpha}(f|x))^2|x) \quad (44)$$

where $p_{\mathcal{S},f}^{\alpha}(f|x) \equiv \sum_y p_{\mathcal{S},f}^{\alpha}(y|x)f(x, y)$

(c) For each $f \in \mathcal{F}$, update $\alpha(f)$ by

$$\alpha(f) \leftarrow \alpha(f) + \frac{1}{r(f)} \log \left(1 - \frac{1}{r(f)} \frac{G'(f)}{G''(f)} \right) \quad (45)$$

6. For each $f \in \mathcal{F}$, substitute $\alpha(f)$ into (26) to determine $\sim\Delta L(\mathcal{S}, f)$.

Convergence for this algorithm is guaranteed just as it was for algorithm 3 – after each iteration of step 5, the value of $\alpha(f)$ for each candidate feature f is closer to its optimal value $\alpha^*(f)$ and, more importantly, the gain $G_{\mathcal{S},f}$ is closer to the maximal gain $\sim\Delta L(\mathcal{S}, f)$.

Alpino: Wide-coverage Computational Analysis of Dutch

Gosse Bouma, Gertjan van Noord, and Robert Malouf

Alfa-informatica
Rijksuniversiteit Groningen

Abstract

Alpino is a wide-coverage computational analyzer of Dutch which aims at accurate, full, parsing of unrestricted text. We describe the head-driven lexicalized grammar and the lexical component, which has been derived from existing resources. The grammar produces dependency structures, thus providing a reasonably abstract and theory-neutral level of linguistic representation. An important aspect of wide-coverage parsing is robustness and disambiguation. The dependency relations encoded in the dependency structures have been used to develop and evaluate both hand-coded and statistical disambiguation methods.

1 Introduction

For English, tremendous progress has been made in the area of wide-coverage parsing of unrestricted text. Many of the proposed systems are statistical parsers, but systems based on a hand-written grammar exist as well. The aim of Alpino¹ is to provide computational analysis of Dutch with coverage and accuracy comparable to state-of-the-art parsers for English.

The Alpino grammar (described in more detail below) is a lexicalized grammar in the tradition of constructionalist Head-driven Phrase Structure Grammar (Pollard and Sag 1994, Sag 1997). The grammar consists of hand-written, linguistically motivated rules and lexical types. To evaluate the coverage and disambiguation component of the system, a testbench of syntactically annotated material is absolutely crucial. Given the current lack of such material for Dutch, we have started to annotate corpora with dependency structures. Dependency structures provide a convenient level of representation for annotation, and a fairly neutral representation for further processing. The annotation format is taken from the project *Corpus Gesproken Nederlands (Corpus of Spoken Dutch)* (Oostdijk 2000). The construction of dependency structures in the grammar and our treebanking efforts are described in section 4. Both the lexicalist nature of the Alpino grammar and the use of dependency structures imply that lexical items must be associated with detailed valency information. For the Alpino lexicon we have extracted this information from the Celex and Parole lexical databases (section 3).

In section 5 we describe Alpino's parsing architecture. Section 6 describes a variety of disambiguation strategies which have been integrated in Alpino. In addition, we report on a number of preliminary disambiguation experiments. We conclude with some remarks on future work.

¹Alpino is being developed as part of the NWO PIONIER project *Algorithms for Linguistic Processing*, www.let.rug.nl/~vannoord/alp

2 Grammar

The Alpino grammar is an extension of the successful OVIS grammar (van Noord, Bouma, Koeling and Nederhof 1999, Veldhuijzen van Zanten, Bouma, Sima'an, van Noord and Bonnema 1999), a lexicalized grammar in the tradition of Head-driven Phrase Structure Grammar (Pollard and Sag 1994). The grammar formalism is carefully designed to allow linguistically sophisticated analyses as well as efficient and robust processing.

In contrast to earlier work on HPSG grammar rules in Alpino are relatively detailed. However, as pointed out in Sag (1997), by organizing rules in an inheritance hierarchy, the relevant linguistic generalizations can still be captured. The Alpino grammar currently contains over 100 rules, defined in terms of a few general rule structures and principles. The grammar covers the basic constructions of Dutch (including main and subordinate clauses, (indirect) questions, imperatives, (free) relative clauses, a wide range of verbal and nominal complementation and modification patterns, and coordination) as well as a wide variety of more idiosyncratic constructions (appositions, verb-particle constructions, PP's including a particle, NP's modified by an adverb, punctuation, etc.). The lexicon contains definitions for various nominal types (nouns with various complementation patterns, proper names, pronouns, temporal nouns, deverbalized nouns), various complementizer, determiner, and adverb types, adjectives, and 36 verbal subcategorization types.

The formalism supports the use of recursive constraints over feature-structures (using delayed evaluation, van Noord and Bouma (1994)). This allowed us to incorporate an analysis of cross-serial dependencies based on argument-inheritance (Bouma and van Noord 1998) and a trace-less account of extraction along the lines of Bouma, Malouf and Sag (2001).

3 Lexical Resources

Accurate, wide-coverage parsing of unrestricted text requires a lexical component with detailed subcategorization frames. For lexicalist grammar formalisms, the availability of lexical resources which specify subcategorization frames is even more crucial. In HPSG, for instance, phrase structure rules rely on the fact that each head contains a specification of the elements it subcategorizes for. If such specifications are missing, the grammar will wildly overgenerate.

We have used two existing lexical databases (Celex and Parole) to create a wide-coverage lexicon with detailed subcategorization frames enriched with dependency relations. Celex (Baayen, Piepenbrock and van Rijn 1993) is a large lexical database for Dutch, with rich phonological and morphological information. For use within the CGN project, this database has been extended with dependency frames (Groot 2000). This version of the lexicon contains 11,800 verbal stems, with a total of 21,800 dependency frames. By far the most frequent frames are those for intransitive (4,100) and transitive (6,500) verbs. A fair number of frames occurs more than 100 times, but 300 of the 650 different dependency frame *types* in the database occur only once.

Dependency Frame	Overlap	Celex only	Parole only	Total
[SU:NP][OBJ1:NP]	1810	1211	240	3261
[SU:NP]	257	1697	42	1996
[SU:NP][PC:PP< <i>pform</i> >]	337	541	273	1151
[SU:NP][OBJ1:NP][PC:PP< <i>pform</i> >]	129	375	308	812
[SU:NP][VC:S<subordinate>]	103	136	103	342
[SUP:NP<het>][OBJ1:NP][SU:CP]	7	247	5	259
[SU:NP][OBJ2:NP][OBJ1:NP]	65	171	28	264
[SU:NP][SE:NP][PC:PP< <i>pform</i> >]	65	62	102	229
[SU:NP][SE:NP]	49	137	65	251
[SU:NP][VC:VP]	10	16	37	63

Table 1: Dependency Frames and the number of stems occurring with this frame in both resources, in CGN/Celex only, in Parole only, and the total number of stems with this dependency frame in the Alpino Lexicon.

The Dutch Parole lexicon² comes with detailed subcategorization information, including dependency relations. The Parole lexicon is smaller than Celex, with 3,200 verbal stems and a total of 5000 dependency frames. There are 320 different dependency frame types, 190 of which occur only once.

Dependency frames for the Alpino lexicon have been constructed using the dependency information provided by CGN/Celex, Parole, and by entering definitions by hand. The latter has been done mostly for auxiliary and modal verbs: a small class of high-frequent elements which are exceptional in a number of ways. The CGN/Celex dictionary is very large. As the Celex database comes with frequency information, we currently only include those lexical items whose frequency is above a certain threshold. For verbal stems, this means that roughly 50% of the stems in Celex is included in the Alpino lexicon. All verbal stems from the Parole lexicon with a dependency frame covered by the grammar are included.

Currently, for 28 different CGN/Celex dependency frames a definition in the grammar has been provided. This covers over 80% of the verbal dependency frames in the CGN/Celex database, 10,400 of which are sufficiently frequent to be included in the Alpino lexicon. For 15 different dependency frames in the Parole lexicon a definition in Alpino is present. Using these, we extract over 4,100 dependency frames (82% of the total number of dependency frames in the Parole database). An overview of overlap and non-overlap for the most frequent frames extractable from both sources is given in table 1. For transitive and intransitive verbs, we see that over 85% of the stems in Parole are present in CGN/Celex as well. For most other dependency frames, however, the overlap is generally much smaller, and a significant portion of the stems present in Parole is not present in

²<http://www.inl.nl/corp/parole.htm>

Celex. This suggests that, for more specific subcategorization frames, both resources are only partially complete, and that not even the union of both provides exhaustive coverage.³

4 Dependency Structures

Within the CGN-project (Oostdijk 2000), guidelines have been developed for syntactic annotation of spoken Dutch (Moortgat, Schuurman and van der Wouden 2000), using dependency structures similar to those used for the German Negra corpus (Skut, Krenn and Uszkoreit 1997).

Dependency structures make explicit the dependency relations between constituents in a sentence. Each non-terminal node in a dependency structure consists of a head-daughter and a list of non-head daughters, whose dependency relation to the head is marked. A dependency structure for (1) is given in figure 1. Control relations are encoded by means of co-indexing (i.e. the subject of *hebben* is the dependent with index **1**). Note that a dependency structure does not necessarily reflect (surface) syntactic constituency. The dependent *haar nieuwe model gisteren aangekondigd*, for instance, does not correspond to a (surface) syntactic constituent in (1).

- (1) Mercedes zou haar nieuwe model gisteren hebben aangekondigd
 Mercedes should her new model yesterday have announced
Mercedes should have announced her new model yesterday

The Alpino grammar produces dependency structures compatible with the CGN-guidelines. We believe this is a useful output format for a number of reasons. First of all, annotating a text with dependency structures is relatively straightforward and independent of the particular grammatical framework assumed. Thus, a dependency treebank can be used to debug and test various versions of the Alpino grammar. Second, as we adopt the CGN-guidelines, a considerable amount of annotated material will be available within the near future which can be used for development and testing. Third, it has been suggested that dependency relations provide a convenient level of representation for evaluation of computational grammar based on radically different grammatical theories (Carroll, Briscoe and Sanfilippo 1998). Finally, statistics for dependency relations between head words can be used to develop accurate models for parse-selection (Collins 1999); preliminary experiments are described in section 6.

Grammatical Construction of Dependency Structures. To produce dependency structures with the Alpino grammar, a new level of representation has been added to the grammar. The attribute DT dominates a dependency structure, with attributes for the lexical head (HD) and the various dependents. The value of a dependent attribute can be a dependency structure or a leaf node consisting of a

³The less frequent verb stems in Celex (currently not included in Alpino) are almost exclusively assigned the intransitive or transitive dependency frame.

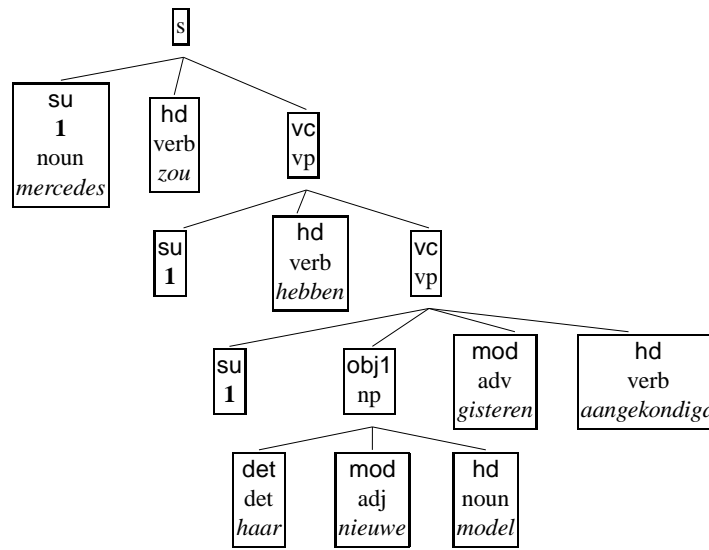


Figure 1: Dependency structure for example (1).

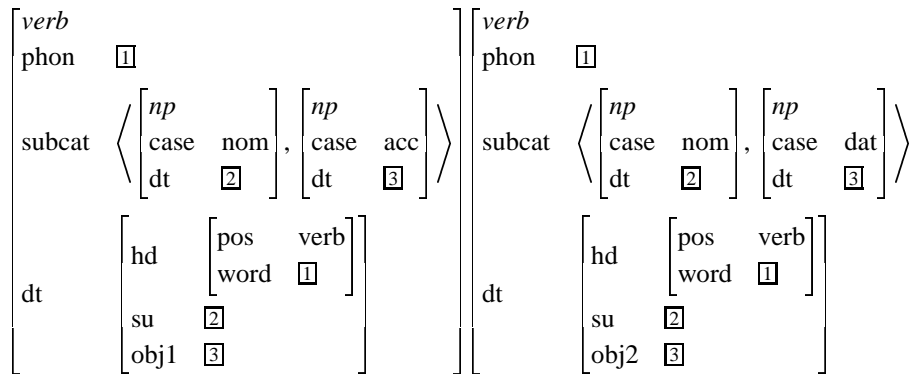


Figure 2: Schematic lexical entry for transitive verbs taking a direct object (OBJ1), and for transitive verbs taking an indirect object (OBJ2).

POS-tag and word only.

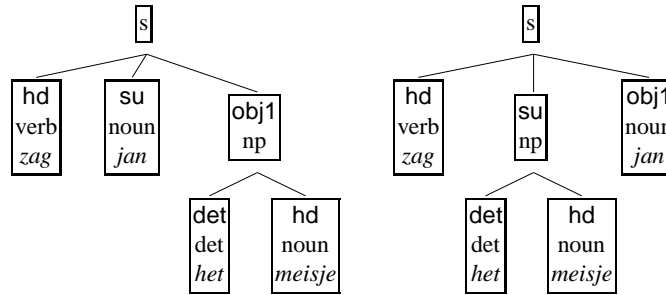
The construction of dependency structures is driven by the lexicon. For each subcategorization type recognized in the lexical hierarchy a mapping between elements on the list-valued feature which specifies basic subcategorization properties (SUBCAT) and attributes of DT is defined. Two examples are given in figure 2. The leftmost feature structure exemplifies a finite, transitive verb. The value of DT of the nominative NP on subcat is identical to the value of the SU dependent. Similarly, the value of DT of the accusative NP on subcat is identical to the value of the OBJ1 dependent. The rightmost feature structure exemplifies a finite, transitive verb for which the object is assigned to the OBJ2 (secondary object) dependency relation. In some cases, the addition of dependency structures leads to more fine-grained distinctions. For instance, PP-arguments can be linked to PC (*prepositional complement*) or LD (*locative or directional complement*), where the distinction between these two is primarily semantic in nature. Therefore, verbs taking a prepositional complement are assigned a subcategorization frame that differs from the frame assigned to verbs taking such a LD complement.

In HEAD-COMPLEMENT structures, the DT attribute can simply be shared between head daughter and mother. In HEAD-MODIFIER structures, the dependency structure of the modifier is added to the list-valued MOD dependent of the head.

Dependency Treebanks. For development and evaluation purposes, we have started to annotate various sample text fragments with dependency structures.

The annotation process typically starts by parsing a sentence with the Alpino grammar. This produces a (often large) number of possible analyses. The annotator picks the analysis which best matches the correct analysis. To facilitate selection of the best parse among a large number of possibilities, the HDRUG environment has been extended with a graphical tool based on the SRI TreeBanker (Carter 1997) which displays all fragments of the input which are a source of ambiguity. By disambiguating these items (usually a much smaller number than the number of readings), the annotator can quickly pick the most accurate parse.

For example, the sentence *Jan zag het meisje* ‘Jan saw the girl’ has (in principle) two readings corresponding to the dependency structures in figure 3. The readings of a sentence are represented as a set of sets of dependency paths, as in figure 4. From these sets of paths, the parse selection tool computes a set of *maximal discriminants* which can be used to select among different analyses. In this case, the path ‘s:hd = v zag’ is shared by all the analyses and so is not a useful discriminant. On the other hand, the path ‘s:obj1:hd = n meisje’ does distinguish between the readings but it is not maximal, since it is subsumed by the path ‘s:obj1 = np het meisje’ which is shorter and makes exactly the same distinctions. The maximal discriminants are presented to the annotator, who may mark any of them as either good (the correct parse must include it) or bad (the correct parse may not include it). In this simple example, marking any one of the maximal discriminants as good or bad is sufficient to uniquely identify the correct parse. For more complex sentences, several choices will have to be made to select a single best parse. To help the annotator, when a discriminant is marked as bad or good, the following

Figure 3: Dependency structures for two readings of *Jan zag het meisje*.

s:hd = v zag	s:hd = v zag
*s:su = np jan	*s:su = np het meisje
*s:obj1 = np het meisje	s:su:det = det het
s:obj1:det = det het	s:su:hd = n meisje
s:obj1:hd = n meisje	*s:obj1 = np jan

Figure 4: Dependency paths for *Jan zag het meisje* (* indicates a maximal discriminant).

inference rules are applied to further narrow the possibilities (Carter 1997):

- If a discriminant is bad, any parse which includes it is bad.
- If a discriminant is good, any parse which does not include it is bad.
- If a discriminant is only included in bad parses, it must be bad.
- If a discriminant is included in all the undecided parses, it must be good.

This allows users to focus their attention on discriminants about which they have clear intuitions. Their decisions about these discriminants combined with the rules of inference can then be used to automatically make decisions about less obvious discriminants.

If the parse selected by the annotator is fully correct, the dependency structure for that parse is stored as XML in the treebank. If the best parse produced by the grammar is not the correct parse as it should be included in the treebank, the dependency structure for this parse is sent to the Thistle editor.⁴ The annotator can now produce the correct parse manually.

We have started to annotate various smaller fragments using the annotation tools described above. The largest fragments consist of two sets of sentences ex-

⁴LT Thistle (Calder 2000), www.ltg.ed.ac.uk/software/thistle/, is an editor and display engine for linguistic data-structures which supports XML.

tracted from the Eindhoven corpus (Uit den Boogaart 1975). The CDBL10 treebank currently consists of the first 519 sentences of ten words or less from section CDBL (newspaper text). The CDBL20 treebank consists of the first 252 sentences with more than 10 but no more than 20 words.

Evaluation. Evaluation of coverage and accuracy of a computational grammar usually is based on some metric which compares tree structures (such as recall and precision of (labelled) brackets or bracketing inconsistencies (crossing brackets) between test item and parser output). As is well-known, such metrics have a number of drawbacks. Therefore, Carroll et al. (1998) propose to annotate sentences with triples of the form $\langle \text{head-word}, \text{dependency relation}, \text{dependent head-word} \rangle$. For instance, for the example in (1) we might obtain:

$\langle \text{zou, su, mercedes} \rangle$	$\langle \text{aangekondigd, obj1, model} \rangle$
$\langle \text{hebben, su, mercedes} \rangle$	$\langle \text{model, det, haar} \rangle$
$\langle \text{aangekondigd, su, mercedes} \rangle$	$\langle \text{model, mod, nieuwe} \rangle$
$\langle \text{aangekondigd, mod, gisteren} \rangle$	

Dependency relations between head-words can be extracted easily from the dependency structures in our treebank, as well as from the dependency structures constructed by the parser. It is thus straightforward to compute precision, recall, and f-score on the set of dependency triples.

5 Robust Parsing

The initial design and implementation of the Alpino parser is inherited from the system described in van Noord (1997), van Noord et al. (1999) and van Noord (2001). However, a number of improvements have been implemented which are described below.

The construction of a dependency structure on the basis of some input proceeds in a number of steps, described below. The first step consists of lexical analysis. In the second step a parse forest is constructed. The third step consists of the selection of the best parse from the parse forest.

Lexical Analysis. The lexicon associates a word or a sequence of words with one or more *tags*. Such tags contain information such as part-of-speech, inflection as well as a subcategorization frame. For verbs, the lexicon typically hypothesizes many different tags, differing mainly in the subcategorization frame. For sentence (1), the lexicon produces 83 tags. Some of those tags are obviously wrong. For example, one of the tags for the word hebben is `verb(hebben,pl,part_sbar_transitive(door))`. The tag indicates a finite plural verb which requires a separable prefix `door`, and which subcategorizes for an SBAR complement. Since `door` does not occur anywhere in sentence (1), this tag will not be useful for this sentence. A filter containing a number of hand-written rules has been implemented which checks that such simple condi-

tions hold. For sentence (1), the filter removes 56 tags. After the filter has applied, feature structures are associated with each of these tags. Often, a single tag is mapped to multiple feature structures. The remaining 27 filtered tags give rise to 89 feature structures.

An important aspect of lexical analysis is the treatment of unknown words. The system applies a number of heuristics for unknown words. Currently, these heuristics attempt to deal with numbers and number-like expressions, capitalized words, words with missing diacritics, words with ‘too many’ diacritics, compounds, and proper names.

If such heuristics still fail to provide an analysis, then the system guesses a tag by inspecting the suffix of the word. A list of suffixes is maintained which predict the tag of a given word. If this still does not provide an analysis, then it is assumed that the word is a noun.

In addition to the treatment of unknown words, the robustness of the system is enhanced by the possibility to skip tokens of the input. Currently this possibility is employed only for certain punctuation marks. Even though punctuation is treated both in the lexicon and the grammar, the syntax of punctuation is irregular enough to warrant the possibility to ignore punctuation. For instance, quotation marks may appear almost anywhere in the input. The corpus contains:

- (2) De z.g. ” speelstraat , die hier en daar al bestaat ?
The so-called ” play-street , that here and there already exists ?

Apparently, the author intended to place `speelstraat` within quotes, but the second quote is not present. During lexical analysis, tags are optionally extended to include neighbouring words which are classified as ‘skipable’.

Creating Parse Forests. The Alpino parser takes the result of lexical analysis as its input, and produces a *parse forest*: a compact representation of all parse trees. The Alpino parser is a left-corner parser with selective memoization and goal-weaking. It is a variant of the parsers described in van Noord (1997). We generalized some of the techniques described there to take into account relational constraints, which are delayed until sufficiently instantiated (van Noord and Bouma 1994).

As described in van Noord et al. (1999) and van Noord (2001), the parser can be instructed to find all occurrences of the start category *anywhere in the input*. This feature is added to enhance robustness as well. In case the parser cannot find an instance of the start category from the beginning of the sentence to the end, then the parser produces parse trees for large chunks of the input. A best-first search procedure then picks out the best sequence of such chunks. Depending on the application, such chunks might be very useful. In the past, we successfully employed this strategy in a spoken dialogue system (Veldhuijzen van Zanten et al. 1999).

beam	cdb110		cdb120	
	accuracy (%)	speed (msec)	accuracy (%)	speed (msec)
1	79.99	190	73.63	740
2	80.66	270	74.59	1470
4	81.11	350	75.07	2350
8	81.22	530	75.35	3630
16	81.36	590	75.31	5460
32	81.36	790	74.98	7880
∞	81.36	640	-	-

Table 2: Effect of beam-size on accuracy and efficiency of parse selection

Unpacking and Parse Selection. The motivation to construct a parse forest is efficiency: the number of parse trees for a given sentence can be enormous. In addition to this, in most applications the objective will not be to obtain *all* parse trees, but rather the *best* parse tree. Thus, the final component of the parser consists of a procedure to select these best parse trees from the parse forest.

In order to select the best parse tree from a parse forest, we assume a parse evaluation function which assigns a score to each parse. In section 6 we describe some initial experiments with a variety of parse evaluation functions. A naive algorithm constructs all possible parse trees, assigns each one a score, and then selects the best one. Since it is too inefficient to construct all parse trees, we have implemented the algorithm which computes parse trees from the parse forest as a best-first search. This requires that the parse evaluation function is extended to partial parse trees. In order to be able to *guarantee* that this search procedure indeed finds the best parse tree, a certain monotonicity requirement should apply to this evaluation function: if a (partial) tree s is better than s' , then a tree t which contains s should be better than t' which is just like t except it has s' instead of s . However, instead of relying on such a requirement, we implemented a variant of a best-first search algorithm in such a way that for each state in the search space, we maintain the b best candidates, where b is a small integer (the *beam*). If the beam is decreased, then we run a larger risk of missing the best parse (but the result will typically still be a relatively ‘good’ parse); if the beam is increased, then the amount of computation increases too. Currently, we find that a value of $b = 4$ is a good compromise between accuracy and efficiency. In table 2 the effect of various values for b is presented for two development treebanks. The grammar assigns on average about 33 parse trees per sentence for the cdb110 corpus. This number increases rapidly for longer sentences: for the cdb120 corpus it is at least 340.⁵

⁵This is the average number after creating all parse trees for each sentence with a maximum of 1000 parse trees per sentence.

6 Disambiguation

The best-first unpack strategy described in section 5 depends on a parse evaluation function which assigns scores to (partial) parse trees. We have experimented with a number of disambiguation techniques on the `cdb110` and `cdb120` development treebanks described earlier.

Penalty rules. The simplest disambiguation method consists of hand-written ‘penalty’ rules which implement a variety of preferences. Each such penalty rule describes a partial parse tree. For a given parse tree, the system computes how often a sub-tree matches with a penalty rule, giving rise to the total penalty of that parse. The following lists characterizes some of the penalty rules:

- complementation is preferred over modification
- subject topicalization is preferred over object topicalization
- long distance dependencies are dis-preferred
- certain rules are dis-preferred (e.g. rules which coordinate categories without an explicit coordinator)
- certain lexical entries are dis-preferred (e.g. the preposition readings for the words *aan*, *bij*, *in*, *naar*, *op*, *uit*, *voor*, *tussen* are preferred over the adjectival, noun and/or verb readings).
- certain guesses for unknown words are preferred over others

As can be concluded from the preliminary results presented in table 3, it appears to be the case that about 60% of the disambiguation problem can be solved using this very simple technique.

Dependency relations We also experimented with statistical models based on dependency relations encoded in the dependency structure. The model assigns a probability to a parse by considering each dependency relation. For this purpose, dependency relations d are 5-tuples $d = \langle w_h, p_h, r, w_a, p_a \rangle$ where w_h is the head word, p_h is the corresponding part-of-speech tag taken from a small set of part-of-speeches $\{v, n, a, adv, p, \dots\}$, r is the name of the relation taken from a small set of relation names $\{su, obj1, obj2, vc, mod, det, \dots\}$; w_a is the argument word, and p_a is its part of speech.

The probability of a parse y given a sentence x might then be defined as:

$$p(y|x) = \frac{1}{Z(x)} \prod_{d \in y} p(r, w_a, p_a | w_h, p_h)$$

For disambiguation, the normalizing factor $Z(x)$ is the same for every parse of a given sentence and can be ignored.

Due to the occurrence of reentrancies, dependency structures are generally not trees but graphs. Therefore, the product above gives poor results because it will have an unjustified bias against such reentrancies (a reentrancy gives rise to an additional dependency relation). For this reason, we have chosen to score parse trees by determining the *mean* value of $-\log p$ for each tuple; this improved results considerably. The probability of a dependency is calculated as follows:

$$p(r, w_a, p_a | w_h, p_h) = p(r | w_h, p_h) * p(p_a | w_h, p_h, r) * p(w_a | w_h, w_p, r, p_a)$$

The three components are each calculated using a linear back-off strategy, where the weights are determined by frequency and diversity (formula 2.66 of (Collins 1999)). The quantities we use for backing off are given in the following table:

back-off level	$p(r w_h, p_h)$	$p(p_a w_h, p_h, r)$	$p(w_a w_h, w_p, r, p_a)$
1	$p(r p_h)$	$p(p_a p_h, r)$	$p(w_a w_p, r, p_a)$
2	$p(r)$	$p(p_a r)$	$p(w_a r, p_a)$
3		$p(p_a)$	$p(w_a p_a)$
4			$p(w_a)$

Because the size of the treebanks we have currently available is much too small to estimate these quantities accurately, we have chosen to do our estimation using unsupervised learning. We have parsed a large corpus ('de Volkskrant' newspaper text: first four months of 1997) using the penalty rules described in the previous section as our disambiguator. This corpus contains about 350,000 sentences and 6,200,000 words. We only used those sentences that the system could analyse as a single constituent, and within a reasonable amount of time. This meant that we could use the results of about 225,000 sentences. We estimated the quantity p using the best parse (according to the penalty rules) for each of these sentences. Collecting the 225,000 dependency structures took about one month of CPU-time (using the high-performance computing cluster of the University of Groningen).

As can be concluded from table 3, such a model performs much better than the baseline. Moreover, a combined model in which we simply add the rule penalties to the quantity p performs better than either model in isolation.

Log-linear models. While the model described in the previous section offers good performance and conceptual simplicity, it is not without problems. In particular, the strategies for dealing with reentrancies in the dependency structures and for combining scores derived from penalty rules and from dependency relation statistics are ad hoc. Log-linear models, introduced to natural language processing by Berger, Della Pietra and Della Pietra (1996) and Della Pietra, Della Pietra and Lafferty (1997), and applied to stochastic constraint-based grammars by Abney (1997) and Johnson, Geman, Canon, Chi and Riezler (1999), offer the potential to solve both of these problems. Given a conditional log-linear model, the probability of a sentence x having the parse y is:

$$p(y|x) = \frac{1}{Z(x)} \exp \left(\sum_i \lambda_i f_i(x, y) \right)$$

technique	cdb110			cdb120		
	precision	recall	f-score	precision	recall	f-score
baseline	62.3	63.3	62.8	58.5	59.6	59.0
log linear	76.0	76.6	76.3	66.3	67.6	66.0
penalties	78.6	79.3	78.9	73.1	73.3	73.2
dependency rel's	78.9	79.7	79.3	69.7	71.1	70.4
heur. + dep-rel's	80.9	81.7	81.3	74.6	75.4	75.0
maximum	89.1	90.0	89.6	83.2	84.1	83.7

Table 3: Preliminary results on the cdb110 and cdb120 development treebanks for a number of disambiguation techniques. The *baseline* row lists the percentages obtained if we select for each sentence a random parse tree from the parse forest. The *maximum* row lists the percentages obtained if we take for each sentence the best parse tree. These two numbers thus indicate the lower and upper bounds for parse selection.

As before, the partition function $Z(x)$ will be the same for every parse of a given sentence and can be ignored, so the score for a parse is simply the weighted sum of the property functions $f_i(x,y)$. What makes log-linear models particularly well suited for this application is that the property functions may be sensitive to any information which might be useful for disambiguation. Possible property functions include syntactic heuristics, lexicalized and backed-off dependency relations, structural configurations, and lexical semantic classes. Using log-linear models, all of these disparate types of information may be combined into a single model for disambiguation. Furthermore, since standard techniques for estimating the weights λ_i from training data make no assumptions about the independence of properties, one need not take special precautions when information sources overlap.

The drawback to using log-linear models is that accurate estimation of the parameters λ_i requires a large amount of annotated training data. Since such training data is not yet available, we instead attempted unsupervised training from unannotated data. We used the Alpino parser to find all parses of the 82,000 sentences with ten or fewer words in the ‘de Volkskrant’ newspaper corpus. Using the resulting collection of 2,200,000 unranked parses, we then applied Riezler et al.’s (2000) ‘Iterative Maximization’ algorithm to estimate the parameters of a log-linear model with dependency tuples as described in the previous section as property functions. The results, given in table 3, show some promise, but the performance of the log-linear model does not yet match that of the other disambiguation strategies. Current work in this area is focused on expanding the set of properties and on using supervised training from what annotated data is available to bootstrap the unsupervised training from large quantities of newspaper text.

7 Conclusions

Alpino aims at providing a wide-coverage, accurate, computational grammar for Dutch. The linguistic component of the system consists of a lexicalist feature-based grammar for Dutch, a wide-coverage and detailed lexicon, and a method for constructing dependency treebanks. The parser contains a lexical analysis module and a method for reconstructing parses from a parse forest using beam search, which allows the linguistic knowledge to be applied efficiently and robustly to unrestricted text. Finally, we have presented preliminary experiments aimed at providing accurate disambiguation.

In the near future, we hope to address a number of additional issues. The valency information in the lexicon is in many ways incomplete. We hope to obtain a more complete lexicon by acquiring dependency frames from corpora. Lexical analysis currently uses hand-written filter rules to reduce the number of tags for lexical items. An obvious alternative is to use a corpus-based part-of-speech tagger to arrive at the relevant filters. Finally, the work on disambiguation can profit from the availability of more annotated material. This suggests that our efforts at creating a dependency treebank may lead to improved results in the future.

References

- Abney, S. P.(1997), Stochastic attribute-value grammars, *Computational Linguistics* **23**, 597–618.
- Baayen, R. H., Piepenbrock, R. and van Rijn, H.(1993), *The CELEX Lexical Database (CD-ROM)*, Linguistic Data Consortium, University of Pennsylvania, Philadelphia, PA.
- Berger, A., Della Pietra, S. and Della Pietra, V.(1996), A maximum entropy approach to natural language processing, *Computational Linguistics* **22**(1), 39–72.
- Bouma, G. and van Noord, G.(1998), Word order constraints on verb clusters in German and Dutch, in E. Hinrichs, T. Nakazawa and A. Kathol (eds), *Complex Predicates in Nonderivational Syntax*, Academic Press, New York, pp. 43–72.
- Bouma, G., Malouf, R. and Sag, I.(2001), Satisfying constraints on adjunction and extraction, *Natural Language and Linguistic Theory* **19**, 1–65.
- Calder, J.(2000), Thistle and interarbora, *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, Saarbrücken, pp. 992–996.
- Carroll, J., Briscoe, T. and Sanfilippo, A.(1998), Parser evaluation: A survey and a new proposal, *Proceedings of the first International Conference on Language Resources and Evaluation (LREC)*, Granada, Spain, pp. 447–454.
- Carter, D.(1997), The TreeBanker: A tool for supervised training of parsed corpora, *Proceedings of the ACL Workshop on Computational Environments For Grammar Development And Linguistic Engineering*, Madrid.

- Collins, M.(1999), *Head-Driven Statistical Models for Natural Language Parsing*, PhD thesis, University Of Pennsylvania.
- Della Pietra, S., Della Pietra, V. and Lafferty, J.(1997), Inducing features of random fields, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**, 380–393.
- Groot, M.(2000), Lexiconopbouw: microstructuur. Internal report Corpus Gesproken Nederlands.
- Johnson, M., Geman, S., Canon, S., Chi, Z. and Riezler, S.(1999), Estimators for stochastic “unification-based” grammars, *Proceedings of the 37th Annual Meeting of the ACL*, College Park, Maryland, pp. 535–541.
- Moortgat, M., Schuurman, I. and van der Wouden, T.(2000), CGN syntactische annotatie. Internal report Corpus Gesproken Nederlands.
- Oostdijk, N.(2000), The Spoken Dutch Corpus: Overview and first evaluation, *Proceedings of Second International Conference on Language Resources and Evaluation (LREC)*, pp. 887–894.
- Pollard, C. and Sag, I.(1994), *Head-driven Phrase Structure Grammar*, University of Chicago / CSLI.
- Riezler, S., Prescher, D., Kuhn, J. and Johnson, M.(2000), Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and em, *Proceedings of the 38th Annual Meeting of the ACL*, Hong Kong, pp. 480–487.
- Sag, I.(1997), English relative clause constructions, *Journal of Linguistics* **33**(2), 431–484.
- Skut, W., Krenn, B. and Uszkoreit, H.(1997), An annotation scheme for free word order languages, *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, DC.
- Uit den Boogaart, P. C.(1975), *Woordfrequenties in geschreven en gesproken Nederlands*, Oosthoek, Scheltema & Holkema, Utrecht. Werkgroep Frequentieonderzoek van het Nederlands.
- van Noord, G.(1997), An efficient implementation of the head corner parser, *Computational Linguistics* **23**(3), 425–456. cmp-1g/9701004.
- van Noord, G.(2001), Robust parsing of word graphs, in J.-C. Junqua and G. van Noord (eds), *Robustness in Language and Speech Technology*, Kluwer Academic Publishers, Dordrecht.
- van Noord, G. and Bouma, G.(1994), Adjuncts and the processing of lexical rules, *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, Kyoto, pp. 250–256. cmp-1g/9404011.
- van Noord, G., Bouma, G., Koeling, R. and Nederhof, M.-J.(1999), Robust grammatical analysis for spoken dialogue systems, *Journal of Natural Language Engineering* **5**(1), 45–93.
- Veldhuijzen van Zanten, G., Bouma, G., Sima’an, K., van Noord, G. and Bonnema, R.(1999), Evaluation of the NLP components of the OVIS2 spoken dialogue system, in F. van Eynde, I. Schuurman and N. Schelkens (eds), *Computational Linguistics in the Netherlands 1998*, Rodopi Amsterdam, pp. 213–229.

Inducing Features of Random Fields

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty, *Member, IEEE*

Abstract—We present a technique for constructing random fields from a set of training samples. The learning paradigm builds increasingly complex fields by allowing potential functions, or features, that are supported by increasingly large subgraphs. Each feature has a weight that is trained by minimizing the Kullback-Leibler divergence between the model and the empirical distribution of the training data. A greedy algorithm determines how features are incrementally added to the field and an iterative scaling algorithm is used to estimate the optimal values of the weights.

The random field models and techniques introduced in this paper differ from those common to much of the computer vision literature in that the underlying random fields are non-Markovian and have a large number of parameters that must be estimated. Relations to other learning approaches, including decision trees, are given. As a demonstration of the method, we describe its application to the problem of automatic word classification in natural language processing.

Keywords—Random field, Kullback-Leibler divergence, iterative scaling, maximum entropy, EM algorithm, statistical learning, clustering, word morphology, natural language processing.

I. INTRODUCTION

IN this paper we present a method for incrementally constructing random fields. Our method builds increasingly complex fields to approximate the empirical distribution of a set of training examples by allowing potential functions, or features, that are supported by increasingly large subgraphs. Each feature is assigned a weight, and the weights are trained to minimize the Kullback-Leibler divergence between the field and the empirical distribution of the training data. Features are incrementally added to the field using a top-down greedy algorithm, with the intent of capturing the salient properties of the empirical sample while allowing generalization to new configurations. The general problem that the methods we propose address is that of discovering the structure inherent in a set of sample patterns. As one of the fundamental aims of statistical inference and learning, this problem is central to a wide range of tasks including classification, compression, and prediction.

To illustrate the nature of our approach, suppose we wish to automatically characterize spellings of words according to a statistical model; this is the application we develop in Section 5. A field with no features is simply a uniform distribution on ASCII strings (where we take the distribution of string lengths as given). The most conspicuous feature of English spellings is that they are most commonly comprised of lower-case letters. The induction algorithm makes this observation by first constructing the field

$$p(\omega) = \frac{1}{Z} e^{\sum_i \lambda_{[a-z]} \chi_{[a-z]}(\omega_i)}$$

where χ is an indicator function and the weight $\lambda_{[a-z]}$ associated with the feature that a character is lower-case is chosen to be approximately 1.944. This means that a string with a lowercase letter in some position is about $7 \approx e^{1.944}$ times more likely than

the same string without a lowercase letter in that position. The following collection of strings was generated from the resulting field by Gibbs sampling. (As for all of the examples that will be shown, this sample was generated with annealing, to concentrate the distribution on the more probable strings.)

m, r, xevo, ijjiir, b, to, jz, gsr, wq, vf, x, ga, msmGh, pcp, d, oziVlal, hzagh, yzop, io, advzmxnv, ijv_bolft, x, emx, kayerf, mlj, rawzyb, jp, ag, ctdnnnbg, wgdw, t, kguv, cy, spxcq, uzflbbf, dxtkkn, cxwx, jpd, ztzh, lv, zhpkvnu, l^, r, qee, nynrx, atze4n, ik, se, w, lrh, hp+, yrqyka'h, zcngotcnx, igcump, zjcjs, lqWiqu, cefmfhc, o, lb, fdcY, tzby, yopxmvk, by, fz,, t, govycm, ijyiduwfzo, 6xr, duh, ejv, pk, pjw, l, fl, w

The second most important feature, according to the algorithm, is that two adjacent lower-case characters are extremely common. The second-order field now becomes

$$p(\omega) = \frac{1}{Z} e^{\sum_{i \sim j} \lambda_{[a-z][a-z]} \chi_{[a-z][a-z]}(\omega_{ij}) + \sum_i \lambda_{[a-z]} \chi_{[a-z]}(\omega_i)}$$

where the weight $\lambda_{[a-z][a-z]}$ associated with adjacent lower-case letters is approximately 1.80.

The first 1000 features that the algorithm induces include the strings $\langle s \rangle$, $\langle re \rangle$, $\langle ly \rangle$, and $\langle ing \rangle$, where the character “<” denotes beginning-of-string and the character “>” denotes end-of-string. In addition, the first 1000 features include the regular expressions $[0-9][0-9]$ (with weight 9.15) and $[a-z][A-Z]$ (with weight -5.81) in addition to the first two features $[a-z]$ and $[a-z][a-z]$. A set of strings obtained by Gibbs sampling from the resulting field is shown here:

was, reaser, in, there, to, will, ,, was, by, homes, thing, be, reloverated, ther, which, consists, at, fores, anditing, with, Mr., proveral, the, ,, **, on't, prolling, prothere, ,, mento, at, yaou, l, chestraing, for, have, to, intrally, of, qut, ., best, compers, **, cluseliment, uster, of, is, deveral, this, thise, of, offect, inatever, thifer, constranded, stater, vill, in, thase, in, youse, menttering, and, ., of, in, verate, of, to

These examples are discussed in detail in Section 5.

The induction algorithm that we present has two parts: *feature selection* and *parameter estimation*. The greediness of the algorithm arises in feature selection. In this step each feature in a pool of candidate features is evaluated by estimating the reduction in the Kullback-Leibler divergence that would result from adding the feature to the field. This reduction is approximated as a function of a single parameter, and the largest value of this function is called the *gain* of the candidate. This approximation is one of the key elements of our approach, making it practical to evaluate a large number of candidate features at each stage of the induction algorithm. The candidate with the largest gain is added to the field. In the parameter estimation step, the parameters of the field are estimated using an iterative scaling algorithm. The algorithm we use is a new statistical estimation algorithm

Stephen and Vincent Della Pietra are with Renaissance Technologies, Stony Brook, NY, 11790. E-mail: [sdella,vdella]@rentec.com

John Lafferty is with the Computer Science Department of the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 15213. E-mail: lafferty@cs.cmu.edu

that we call *Improved Iterative Scaling*. It is an improvement of the Generalized Iterative Scaling algorithm of Darroch and Ratcliff [12] in that it does not require that the features sum to a constant. The improved algorithm is easier to implement than the Darroch and Ratcliff algorithm, and can lead to an increase in the rate of convergence by increasing the size of the step taken toward the maximum at each iteration. In Section 4 we give a simple, self-contained proof of the convergence of the improved algorithm that does not make use of the Kuhn-Tucker theorem or other machinery of constrained optimization. Moreover, our proof does not rely on the convergence of alternating I-projection as in Csiszár's proof [10] of the Darroch-Ratcliff procedure.

Both the feature selection step and the parameter estimation step require the solution of certain algebraic equations whose coefficients are determined as expectation values with respect to the field. In many applications these expectations cannot be computed exactly because they involve a sum over an exponentially large number of configurations. This is true of the application that we develop in Section 5. In such cases it is possible to approximate the equations that must be solved using Monte Carlo techniques to compute expectations of random variables. The application that we present uses Gibbs sampling to compute expectations, and the resulting equations are then solved using Newton's method.

Our method can be viewed in terms of the *principle of maximum entropy* [19], which instructs us to assume an exponential form for our distributions, with the parameters viewed as Lagrange multipliers. The techniques that we develop in this paper apply to exponential models in general. We formulate our approach in terms of random fields because this provides a convenient framework within which to work, and because our main application is naturally cast in these terms.

Our method differs from the most common applications of statistical techniques in computer vision and natural language processing. In contrast to many applications in computer vision, which involve only a few free parameters, the typical application of our method involves the estimation of thousands of free parameters. In addition, our methods apply to general exponential models and random fields—there is no underlying Markov assumption made. In contrast to the statistical techniques common to natural language processing, in typical applications of our method there is no probabilistic finite-state or push-down automaton on which the statistical model is built.

In the following section we describe the form of the random field models considered in this paper and the general learning algorithm. In Section 3 we discuss the feature selection step of the algorithm and briefly address cases when the equations need to be estimated using Monte Carlo methods. In Section 4 we present the Improved Iterative Scaling algorithm for estimating the parameters, and prove the convergence of this algorithm. In Section 5 we present the application of inducing features of spellings, and finally in Section 6 we discuss the relation between our methods and other learning approaches, as well as possible extensions of our method.

II. THE LEARNING PARADIGM

In this section we present the basic algorithm for building up a random field from elementary features. The basic idea

is to incrementally construct an increasingly detailed field to approximate a reference distribution \tilde{p} . Typically the distribution \tilde{p} is obtained as the empirical distribution of a set of training examples. After establishing our notation and defining the form of the random field models we consider, we present the training problem as a statement of two equivalent optimization problems. We then discuss the notions of a candidate feature and the gain of a candidate. Finally, we give a statement of the induction algorithm.

A. Form of the random field models

Let $G = (E, V)$ be a finite graph with vertex set V and edge set E , and let \mathcal{A} be a finite alphabet. The *configuration space* Ω is the set of all labelings of the vertices in V by letters in \mathcal{A} . If $C \subset V$ and $\omega \in \Omega$ is a configuration, then ω_C denotes the configuration restricted to C . A *random field* on G is a probability distribution on Ω . The set of all random fields is nothing more than the simplex Δ of all probability distributions on Ω . If $f : \Omega \rightarrow \mathbf{R}$ then the *support* of f , written $\text{supp}(f)$, is the smallest vertex subset $C \subset V$ having the property that whenever $\omega, \omega' \in \Omega$ with $\omega_C = \omega'_C$ then $f(\omega) = f(\omega')$.

We consider random fields that are given by Gibbs distributions of the form

$$p(\omega) = \frac{1}{Z} e^{\sum_C V_C(\omega)}$$

for $\omega \in \Omega$, where $V_C : \Omega \rightarrow \mathbf{R}$ are functions with $\text{supp}(V_C) = C$. The field is *Markov* if whenever $V_C \neq 0$ then C is a *clique*, or totally connected subset of V . This property is expressed in terms of conditional probabilities as

$$p(\omega_u | \omega_v, v \neq u) = p(\omega_u | \omega_v, (u, v) \in E)$$

where u and v are arbitrary vertices. We assume that each C is a path-connected subset of V and that

$$V_C(\omega) = \sum_{1 \leq i \leq n_C} \lambda_i^C f_i^C(\omega) = \lambda^C \cdot f^C(\omega)$$

where $\lambda_i^C \in \mathbf{R}$ and $f_i^C(\omega) \in \{0, 1\}$. We say that the values λ_i^C are the *parameters* of the field and that the functions f_i^C are the *features* of the field. In the following, it will often be convenient to use notation that disregards the dependence of the features and parameters on a vertex subset C , expressing the field in the form

$$p(\omega) = \frac{1}{Z} e^{\sum_i \lambda_i f_i(\omega)} = \frac{1}{Z} e^{\lambda \cdot f(\omega)}.$$

For every random field $(E, V, \{\lambda_i, f_i\})$ of the above form, there is a field $(E', V, \{\lambda_i, f_i\})$ that is Markovian, obtained by completing the edge set E to ensure that for each i , the subgraph generated by the vertex subset $C = \text{supp}(f_i)$ is totally connected.

If we impose the constraint $\lambda_i = \lambda_j$ on two parameters λ_i and λ_j , then we say that these parameters are *tied*. If λ_i and λ_j are tied, then we can write

$$\lambda_i f_i(\omega) + \lambda_j f_j(\omega) = \lambda g(\omega)$$

where $g = f_i + f_j$ is a *non-binary* feature. In general, we can collapse any number of tied parameters onto a single parameter

associated with a non-binary feature. Having tied parameters is often natural for a particular problem, but the presence of non-binary features generally makes the estimation of parameters more difficult.

An *automorphism* σ of a graph is a permutation of the vertices that takes edges to edges: $(u, v) \in E$ if and only if $(\sigma u, \sigma v) \in E$. A random field $(E, V, \{\lambda_i, f_i\})$ is said to have *homogeneous features* if for each feature f_i and automorphism σ of the graph $G = (E, V)$, there is a feature f_j such that $f_j(\sigma\omega) = f_i(\omega)$ for $\omega \in \Omega$. If in addition $\lambda_j = \lambda_i$, then the field is said to be *homogeneous*. Roughly speaking, a homogeneous feature contributes the same weight to the distribution no matter where in the graph it appears. Homogeneous features arise naturally in the application of Section 5.

The methods that we describe in this paper apply to exponential models in general; that is, it is not essential that there is an underlying graph structure. However, it will be convenient to express our approach in terms of the random field models described above.

B. Two optimization problems

Suppose that we are given an initial model $q_0 \in \Delta$, a reference distribution \tilde{p} , and a set of features $f = (f_0, f_1, \dots, f_n)$. In practice, it is often the case that \tilde{p} is the empirical distribution of a set of training samples $\omega^{(1)}, \omega^{(2)} \dots \omega^{(N)}$, and is thus given by

$$\tilde{p}(\omega) = \frac{c(\omega)}{N}$$

where $c(\omega) = \sum_{1 \leq i \leq N} \delta(\omega, \omega^{(i)})$ is the number of times that configuration ω appears among the training samples.

We wish to construct a probability distribution $q_* \in \Delta$ that accounts for these data, in the sense that it approximates \tilde{p} but does not deviate too far from q_0 . We measure distance between probability distributions p and q in Δ using the Kullback-Leibler divergence

$$D(p \| q) = \sum_{\omega \in \Omega} p(\omega) \log \frac{p(\omega)}{q(\omega)}. \quad (1)$$

Throughout this paper we use the notation

$$p[g] = \sum_{\omega \in \Omega} g(\omega) p(\omega)$$

for the expectation of a function $g : \Omega \rightarrow \mathbf{R}$ with respect to the probability distribution p . For a function $h : \Omega \rightarrow \mathbf{R}$ and a distribution q , we use both the notation $h \circ q$ and q_h to denote the generalized Gibbs distribution given by

$$q_h(\omega) = (h \circ q)(\omega) = \frac{1}{Z_q(h)} e^{h(\omega)} q(\omega).$$

Note that $Z_q(h)$ is not the usual partition function. It is a normalization constant determined by the requirement that $(h \circ q)(\omega)$ sums to 1 over ω , and can be written as an expectation:

$$Z_q(h) = q[e^h].$$

There are two natural sets of probability distributions determined by the data \tilde{p} , q_0 , and f . The first is the set $\mathcal{P}(f, \tilde{p})$ of

all distributions that agree with \tilde{p} as to the expected value of the feature function f :

$$\mathcal{P}(f, \tilde{p}) = \{p \in \Delta : p[f] = \tilde{p}[f]\}.$$

The second is the set $\mathcal{Q}(f, q_0)$ of generalized Gibbs distributions based on q_0 with feature function f :

$$\mathcal{Q}(f, q_0) = \{(\lambda \cdot f) \circ q_0 : \lambda \in \mathbf{R}^n\}.$$

We let $\bar{\mathcal{Q}}(f, q_0)$ denote the closure of $\mathcal{Q}(f, q_0)$ in Δ (with respect to the topology it inherits as a subset of Euclidean space).

There are two natural criteria for choosing an element q_* from these sets:

- *Maximum Likelihood Gibbs Distribution.* Choose q_* to be a distribution in $\bar{\mathcal{Q}}(f, q_0)$ with maximum likelihood with respect to \tilde{p} :

$$q_*^{\text{ML}} = \arg \min_{q \in \bar{\mathcal{Q}}(f, q_0)} D(\tilde{p} \| q)$$

- *Maximum Entropy Constrained Distribution.* Choose q_* to be a distribution in $\mathcal{P}(f, \tilde{p})$ that has maximum entropy relative to q_0 :

$$q_*^{\text{ME}} = \arg \min_{p \in \mathcal{P}(f, \tilde{p})} D(p \| q_0)$$

Although these criteria are different, they determine the same distribution: $q_* = q_*^{\text{ML}} = q_*^{\text{ME}}$. Moreover, this distribution is the unique element of the intersection $\mathcal{P}(f, \tilde{p}) \cap \bar{\mathcal{Q}}(f, q_0)$, as we discuss in detail in Section 4.1 and Appendix A.

When \tilde{p} is the empirical distribution of a set of training examples $\omega^{(1)}, \omega^{(2)} \dots \omega^{(N)}$, minimizing $D(\tilde{p} \| p)$ is equivalent to maximizing the probability that the field p assigns to the training data, given by

$$\prod_{1 \leq i \leq N} p(\omega^{(i)}) = \prod_{\omega \in \Omega} p(\omega)^{c(\omega)} \propto e^{-ND(\tilde{p} \| p)}.$$

With sufficiently many parameters it is a simple matter to construct a field for which $D(\tilde{p} \| p)$ is arbitrarily small. This is the classic problem of *over training*. The idea behind the method proposed in this paper is to incrementally construct a field that captures the salient properties of \tilde{p} by incorporating an increasingly detailed collection of features, allowing generalization to new configurations; the resulting distributions are *not* absolutely continuous with respect to the empirical distribution of the training sample. The maximum entropy framework for parameter estimation tempers the over training problem; however, the basic problem remains, and is out of the scope of the present paper. We now present the random field induction paradigm.

C. Inducing field interactions

We begin by supposing that we have a set of *atomic* features

$$\mathcal{F}_{\text{atomic}} \subset \{g : \Omega \rightarrow \{0, 1\}, \text{supp}(g) = v_g \in V\}$$

each of which is supported by a single vertex. We use atomic features to incrementally build up more complicated features. The following definition specifies how we shall allow a field to be incrementally constructed, or *induced*.

Definition 1: Suppose that the field q is given by $q = (\lambda \cdot f) \circ q_0$. The features f_i are called the *active* features of q . A feature g is a *candidate* for q if either $g \in \mathcal{F}_{\text{atomic}}$, or if g is of the form $g(\omega) = a(\omega)f_i(\omega)$ for an atomic feature a and an active feature f_i with $\text{supp}(g) \ominus \text{supp}(f_i) \in E$. The set of candidate features of q is denoted $\mathcal{C}(q)$.

In other words, candidate features are obtained by conjoining atomic features with existing features. The condition on supports ensures that each feature is supported by a path-connected subset of G .

If $g \in \mathcal{C}(q)$ is a candidate feature of q , then we call the 1-parameter family of random fields $q_{\alpha g} = (\alpha g) \circ q$ the *induction* of q by g . We also define

$$G_q(\alpha, g) = D(\tilde{p} \| q) - D(\tilde{p} \| q_{\alpha g}). \quad (2)$$

We think of $G_q(\alpha, g)$ as the improvement that feature g brings to the model when it has weight α . As we show in the following section, $G_q(\alpha, g)$ is \cap -convex in α . (We use the suggestive notation \cap -convex and \cup -convex in place of the less mnemonic concave and convex terminology.) We define $G_q(g)$ to be the greatest improvement that feature g can give to the model while keeping all of the other features' parameters fixed:

$$G_q(g) = \sup_{\alpha} G_q(\alpha, g).$$

We refer to $G_q(g)$ as the *gain* of the candidate g .

D. Incremental construction of random fields

We can now describe our algorithm for incrementally constructing fields.

Field Induction Algorithm.

Initial Data:

A reference distribution \tilde{p} and an initial model q_0 .

Output:

A field q_* with active features f_0, \dots, f_N such that $q_* = \arg \min_{q \in \mathcal{Q}(f, q_0)} D(\tilde{p} \| q)$.

Algorithm:

(0) Set $q^{(0)} = q_0$.

(1) For each candidate $g \in \mathcal{C}(q^{(n)})$ compute the gain $G_{q^{(n)}}(g)$.

(2) Let $f_n = \arg \max_{g \in \mathcal{C}(q^{(n)})} G_{q^{(n)}}(g)$ be the feature with the

largest gain.

(3) Compute $q_* = \arg \min_{q \in \mathcal{Q}(f, q_0)} D(\tilde{p} \| q)$, where $f =$

(f_0, f_1, \dots, f_n) .

(4) Set $q^{(n+1)} = q_*$ and $n \leftarrow n + 1$, and go to step (1).

This induction algorithm has two parts: *feature selection* and *parameter estimation*. Feature selection is carried out in steps (1) and (2), where the feature yielding the largest gain is incorporated into the model. Parameter estimation is carried out in step (3), where the parameters are adjusted to best represent the reference distribution. These two computations are discussed in more detail in the following two sections.

III. FEATURE SELECTION

The feature selection step of our induction algorithm is based upon an approximation. We approximate the improvement due to adding a single candidate feature, measured by the reduction in Kullback-Leibler divergence, by adjusting only the weight of the candidate and keeping all of the other parameters of the field fixed. In general this is only an estimate, since it may well be that adding a feature will require significant adjustments to all of the parameters in the new model. From a computational perspective, approximating the improvement in this way can enable the simultaneous evaluation of thousands of candidate features, and makes the algorithm practical. In this section we present explain the feature selection step in detail.

Proposition 1: Let $G_q(\alpha, g)$, defined in (2), be the approximate improvement obtained by adding feature g with parameter α to the field q . Then if g is not constant, $G_q(\alpha, g)$ is strictly \cap -convex in α and attains its maximum at the unique point $\hat{\alpha}$ satisfying

$$\tilde{p}[g] = q_{\hat{\alpha}g}[g]. \quad (3)$$

Proof: Using the definition (1) of the Kullback-Leibler divergence we can write

$$\begin{aligned} G_q(\alpha, g) &= \sum_{\omega \in \Omega} \tilde{p}(\omega) \log \frac{Z_q^{-1}(\alpha g) e^{\alpha g(\omega)} q(\omega)}{q(\omega)} \\ &= \sum_{\omega \in \Omega} \tilde{p}(\omega) (\alpha g(\omega) - \log q[e^{\alpha g}]) \\ &= \alpha \tilde{p}[g] - \log q[e^{\alpha g}]. \end{aligned}$$

Thus

$$\begin{aligned} \frac{\partial}{\partial \alpha} G_q(\alpha, g) &= \tilde{p}[g] - \frac{q[ge^{\alpha g}]}{q[e^{\alpha g}]} \\ &= \tilde{p}[g] - q_{\alpha g}[g]. \end{aligned}$$

Moreover,

$$\begin{aligned} \frac{\partial^2}{\partial \alpha^2} G_q(\alpha, g) &= \frac{q[ge^{\alpha g}]^2}{q[e^{\alpha g}]^2} - \frac{q[g^2 e^{\alpha g}]}{q[e^{\alpha g}]} \\ &= -q_{\alpha g}[(g - q_{\alpha g}[g])^2] \end{aligned}$$

Hence, $\frac{\partial^2}{\partial \alpha^2} G_q(\alpha, g) \leq 0$, so that $G_q(\alpha, g)$ is \cap -convex in α . If g is not constant, then $\frac{\partial^2}{\partial \alpha^2} G_q(\alpha, g)$, which is minus the variance of g with respect to $q_{\alpha g}$, is strictly negative, so that $G_q(\alpha, g)$ is strictly convex. \square

When g is binary-valued, its gain can be expressed in a particularly nice form. This is stated in the following proposition, whose proof is a simple calculation.

Proposition 2: Suppose that the candidate g is binary-valued. Then $G_q(\alpha, g)$ is maximized at

$$\hat{\alpha} = \log \left(\frac{\tilde{p}[g](1 - q[g])}{q[g](1 - \tilde{p}[g])} \right)$$

and at this value,

$$G_q(g) = G_q(\hat{\alpha}, g) = D(B_p \| B_q)$$

where B_p and B_q are Bernoulli random variables given by

$$\begin{aligned} B_p(1) &= \tilde{p}[g] & B_p(0) &= 1 - \tilde{p}[g] \\ B_q(1) &= q[g] & B_q(0) &= 1 - q[g]. \end{aligned}$$

For features that are not binary-valued, but instead take values in the non-negative integers, the parameter $\hat{\alpha}$ that solves (3) and thus maximizes $G_q(\alpha, g)$ cannot, in general, be determined in closed form. This is the case for tied binary features, and it applies to the application we describe in Section 5. For these cases it is convenient to rewrite (3) slightly. Let $\beta = e^\alpha$ so that $\partial/\partial\alpha = \beta\partial/\partial\beta$. Let

$$g_k = \sum_{\omega} q(\omega) \delta(k, g(\omega))$$

be the total probability assigned to the event that the feature g takes the value k . Then (3) becomes

$$\beta \frac{\partial}{\partial \beta} G_q(\log \beta, g) = \tilde{p}[g] - \frac{\sum_{k=0}^N k g_k \beta^k}{\sum_{k=0}^N g_k \beta^k} = 0 \quad (4)$$

This equation lends itself well to numerical solution. The general shape of the curve $\beta \mapsto \beta\partial/\partial\beta G_q(\log \beta, g)$ is shown in Figure 1.

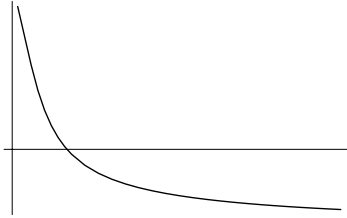


Fig. 1. Derivative of the gain

The limiting value of $\beta\partial G_q(\log \beta, g)/\partial\beta$ as $\beta \rightarrow \infty$ is $\tilde{p}[g] - N$. The solution to equation (4) can be found using Newton's method, which in practice converges rapidly for such functions.

When the configuration space Ω is large, so that the coefficients g_k cannot be calculated by summing over all configurations, Monte Carlo techniques may be used to estimate them. It is important to emphasize that the *same* set of random configurations can be used to estimate the coefficients g_k for each candidate g simultaneously. Rather than discuss the details of Monte Carlo techniques for this problem we refer to the extensive literature on this topic. We have obtained good results using the standard technique of Gibbs sampling [17] for the problem we describe in Section 5.

IV. PARAMETER ESTIMATION

In this section we present an algorithm for selecting the parameters associated with the features of a random field. The algorithm is a generalization of the Generalized Iterative Scaling algorithm of Darroch and Ratcliff [12]. It reduces to the Darroch-Ratcliff algorithm when the features sum to a constant; however, the new algorithm does not make this restriction.

Throughout this section we hold the set of features $f = (f_0, f_1, \dots, f_n)$, the initial model q_0 and the reference distribution \tilde{p} fixed, and we simplify the notation accordingly. In

particular, we write $\gamma \circ q$ instead of $(\gamma \cdot f) \circ q$ for $\gamma \in \mathbf{R}^n$. We assume that $\tilde{p}(\omega) = 0$ whenever $q_0(\omega) = 0$. This condition is commonly written $\tilde{p} \ll q_0$, and it is equivalent to $D(\tilde{p} \| q_0) < \infty$.

A description of the algorithm requires an additional piece of notation. Let

$$f_{\#}(\omega) = \sum_{i=0}^n f_i(\omega).$$

If the features are binary, then $f_{\#}(\omega)$ is the total number of features that are “on” for the configuration ω .

Improved Iterative Scaling.

Initial Data:

A reference distribution \tilde{p} and an initial model q_0 , with $\tilde{p} \ll q_0$, and non-negative features f_0, f_1, \dots, f_n .

Output:

The distribution $q_{\star} = \arg \min_{q \in \mathcal{Q}(f, q_0)} D(\tilde{p} \| q)$

Algorithm:

(0) Set $q^{(0)} = q_0$.

(1) For each i let $\gamma_i^{(k)} \in [-\infty, \infty)$ be the unique solution of

$$q^{(k)}[f_i e^{\gamma_i^{(k)}} f_{\#}] = \tilde{p}[f_i]. \quad (5)$$

(2) Set $q^{(k+1)} = \gamma^{(k)} \circ q^{(k)}$ and $k \leftarrow k + 1$.

(3) If $q^{(k)}$ has converged, set $q_{\star} = q^{(k)}$ and terminate. Otherwise go to step (1).

In other words, this algorithm constructs a distribution $q_{\star} = \lim_{m \rightarrow \infty} \gamma_m \circ q_0$ where $\gamma_m = \sum_{k=0}^m \gamma^{(k)}$ and $\gamma_i^{(k)}$ is determined as the solution to the equation

$$\sum_{\omega} q^{(k)}(\omega) f_i(\omega) e^{\gamma_i^{(k)}} f_{\#}(\omega) = \sum_{\omega} \tilde{p}(\omega) f_i(\omega).$$

When used in the n -th iteration of the field induction algorithm, where a candidate feature $g = f_n$ is added to the field $q = q_n$, we choose the initial distribution q_0 to be $q_0 = q_{\hat{\alpha}g}$, where $\hat{\alpha}$ is the parameter that maximizes the gain of g . In practice, this provides a good starting point from which to begin iterative scaling. In fact, we can view this distribution as the result of applying one iteration of an Iterative Proportional Fitting Procedure [5], [9] to project $q_{\alpha g}$ onto the linear family of distributions with g -marginals constrained to $\tilde{p}[g]$.

Our main result in this section is

Proposition 3: Suppose $q^{(k)}$ is the sequence in Δ determined by the Improved Iterative Scaling algorithm. Then $D(\tilde{p} \| q^{(k)})$ decreases monotonically to $D(\tilde{p} \| q_{\star})$ and $q^{(k)}$ converges to $q_{\star} = \arg \min_{q \in \mathcal{Q}} D(\tilde{p} \| q) = \arg \min_{p \in \mathcal{P}} D(p \| q_0)$.

In the remainder of this section we present a self-contained proof of the convergence of the algorithm. The key idea of the proof is to express the incremental step of the algorithm in terms of an auxiliary function which bounds from below the log-likelihood objective function. This technique is the standard means of analyzing the EM algorithm [13], but it has not previously been applied to iterative scaling. Our analysis of iterative scaling is different and simpler than previous treatments. In particular, in contrast to Csiszár's proof of the Darroch-Ratcliff

procedure [10], our proof does not rely upon the convergence of alternating I-projection [9].

We begin by formulating the basic duality theorem which states that the maximum likelihood problem for a Gibbs distribution and the maximum entropy problem subject to linear constraints have the same solution. We then turn to the task of computing this solution. After introducing auxiliary functions in a general setting, we apply this method to prove convergence of the Improved Iterative Scaling algorithm. We finish the section by discussing Monte Carlo methods for estimating the equations when the size of the configuration space prevents the explicit calculation of feature expectations.

A. Duality

The duality between the maximum likelihood and maximum entropy problems is expressed in the following Proposition.

Proposition 4: Suppose that $\tilde{p} \ll q_0$. Then there exists a unique $q_* \in \Delta$ satisfying

- (1) $q_* \in \mathcal{P} \cap \bar{\mathcal{Q}}$
- (2) $D(p \| q) = D(p \| q_*) + D(q_* \| q)$ for any $p \in \mathcal{P}$ and $q \in \bar{\mathcal{Q}}$
- (3) $q_* = \arg \min_{q \in \bar{\mathcal{Q}}} D(\tilde{p} \| q)$
- (4) $q_* = \arg \min_{p \in \mathcal{P}} D(p \| q_0)$.

Moreover, any of these four properties determines q_* uniquely.

This result is well known, although perhaps not quite in this packaging. In the language of constrained optimization, it expresses the fact that the maximum likelihood problem for Gibbs distributions is the convex dual to the maximum entropy problem for linear constraints. Property (2) is called the *Pythagorean property* since it resembles the Pythagorean theorem if we imagine that $D(p \| q)$ is the square of Euclidean distance and (p, q_*, q) are the vertices of a right triangle.

We include a proof of this result in Appendix A to make this paper self-contained and also to carefully address the technical issues arising from the fact that $\bar{\mathcal{Q}}$ is not closed. The proposition would not be true if we replaced $\bar{\mathcal{Q}}$ with \mathcal{Q} ; in fact, $\mathcal{P} \cap \mathcal{Q}$ might be empty. Our proof is elementary and does not rely on the Kuhn-Tucker theorem or other machinery of constrained optimization.

B. Auxiliary functions

We now turn to the task of computing q_* . Fix \tilde{p} and let $L : \Delta \rightarrow \mathbf{R}$ be the log-likelihood objective function

$$L(q) = -D(\tilde{p} \| q).$$

Definition 2: A function $A : \mathbf{R}^n \times \Delta \rightarrow \mathbf{R}$ is an *auxiliary function* for L if

- (1) For all $q \in \Delta$ and $\gamma \in \mathbf{R}^n$

$$L(\gamma \circ q) \geq L(q) + A(\gamma, q)$$

- (2) $A(\gamma, q)$ is continuous in $q \in \Delta$ and C^1 in $\gamma \in \mathbf{R}^n$ with $A(0, q) = 0$ and

$$\frac{d}{dt} \Big|_{t=0} A(t\gamma, q) = \frac{d}{dt} \Big|_{t=0} L((t\gamma) \circ q).$$

We can use an auxiliary function A to construct an iterative algorithm for maximizing L . We start with $q^{(k)} = q_0$ and recursively define $q^{(k+1)}$ by

$$q^{(k+1)} = \gamma^{(k)} \circ q^{(k)} \quad \text{with} \quad \gamma^{(k)} = \arg \max_{\gamma} A(\gamma, q^{(k)}).$$

It is clear from property (1) of the definition that each step of this procedure increases L . The following proposition implies that in fact the sequence $q^{(k)}$ will reach the maximum of L .

Proposition 5: Suppose $q^{(k)}$ is any sequence in Δ with

$$q^{(0)} = q_0 \quad \text{and} \quad q^{(k+1)} = \gamma^{(k)} \circ q^{(k)}$$

where $\gamma^{(k)} \in \mathbf{R}^n$ satisfies

$$A(\gamma^{(k)}, q^{(k)}) = \sup_{\gamma} A(\gamma, q^{(k)}). \quad (6)$$

Then $L(q^{(k)})$ increases monotonically to $\max_{q \in \bar{\mathcal{Q}}} L(q)$ and $q^{(k)}$ converges to $q_* = \arg \max_{q \in \bar{\mathcal{Q}}} L(q)$.

Equation (6) assumes that the supremum $\sup_{\gamma} A(\gamma, q^{(k)})$ is achieved at finite γ . In Appendix B, under slightly stronger assumptions, we present an extension that allows some components of $\gamma^{(k)}$ to take the value $-\infty$.

To use the proposition to construct a practical algorithm we must determine an auxiliary function $A(\gamma, q)$ for which $\gamma^{(k)}$ satisfying the required condition can be determined efficiently. In Section 4.3 we present a choice of auxiliary function which yields the Improved Iterative Scaling updates.

To prove Proposition 5 we first prove three lemmas.

Lemma 1: If $m \in \Delta$ is a cluster point of $q^{(k)}$, then $A(\gamma, m) \leq 0$ for all $\gamma \in \mathbf{R}^n$.

Proof: Let $q^{(k_l)}$ be a sub-sequence converging to m . Then for any γ

$$\begin{aligned} A(\gamma, q^{(k_l)}) &\leq A(\gamma^{(k_l)}, q^{(k_l)}) \leq L(q^{(k_l+1)}) - L(q^{(k_l)}) \\ &\leq L(q^{(k_l+1)}) - L(q^{(k_l)}). \end{aligned}$$

The first inequality follows from property (6) of $\gamma^{(n_k)}$. The second and third inequalities are a consequence of the monotonicity of $L(q^{(k)})$. The lemma follows by taking limits and using the fact that L and A are continuous. \square

Lemma 2: If $m \in \Delta$ is a cluster point of $q^{(k)}$, then $\frac{d}{dt} \Big|_{t=0} L((t\gamma) \circ m) = 0$ for any $\gamma \in \mathbf{R}^n$.

Proof: By the previous lemma, $A(\gamma, m) \leq 0$ for all γ . Since $A(0, m) = 0$, this means that $\gamma = 0$ is a maximum of $A(\gamma, m)$ so that

$$0 = \frac{d}{dt} \Big|_{t=0} A(t\gamma, m) = \frac{d}{dt} \Big|_{t=0} L((t\gamma) \circ m).$$

\square

Lemma 3: Suppose $\{q^{(k)}\}$ is any sequence with only one cluster point q_* . Then $q^{(k)}$ converges to q_* .

Proof: Suppose not. Then there exists an open set B containing q_* and a subsequence $q^{(n_k)} \notin B$. Since Δ is compact, $q^{(n_k)}$ has a cluster point $q'_* \notin B$. This contradicts the assumption that $\{q^{(k)}\}$ has a unique cluster point. \square

Proof of Proposition 5: Suppose that m is a cluster point of $q^{(k)}$. Then it follows from Lemma 2 that $\frac{d}{dt}|_{t=0} L((t\gamma) \circ m) = 0$, and so $m \in \mathcal{P} \cap \bar{\mathcal{Q}}$ by Lemma 2 of Appendix A. But q_* is the only point in $\mathcal{P} \cap \bar{\mathcal{Q}}$ by Proposition 4. It follows from Lemma 3 that $q^{(k)}$ converges to q_* . \square

In Appendix B we prove an extension of Proposition 5 that allows the components of γ to equal $-\infty$. For this extension, we assume that all the components of the feature function f are non-negative:

$$f_i(\omega) \geq 0 \quad \text{for all } i \text{ and all } \omega. \quad (7)$$

This is not a practical restriction since we can replace f_i by $f_i - \min_{\omega} f_i(\omega)$.

C. Improved Iterative Scaling

We now prove the monotonicity and convergence of the Improved Iterative Scaling algorithm by applying Proposition 5 to a particular choice of auxiliary function. We now assume that each component of the feature function f is non-negative.

For $q \in \Delta$ and $\gamma \in \mathbf{R}^n$, define

$$A(\gamma, q) = 1 + \gamma \cdot \tilde{p}[f] - \sum_{\omega} q(\omega) \sum_i f(i|\omega) e^{\gamma_i f_{\#}(\omega)}$$

where $f(i|\omega) = \frac{f_i(\omega)}{f_{\#}(\omega)}$. It is easy to check that A extends to a continuous function on $(R \cup -\infty)^n \times \Delta$.

Lemma 4: $A(\gamma, q)$ is an extended auxiliary function for $L(q)$. The key ingredient in the proof of the lemma is the \cap -convexity of the logarithm and the \cup -convexity of the exponential, as expressed in the inequalities

$$e^{\sum_i t_i \alpha_i} \leq \sum_i t_i e^{\alpha_i} \quad \text{if } t_i \geq 0, \sum_i t_i = 1 \quad (8)$$

$$\log x \leq x - 1 \quad \text{for all } x > 0. \quad (9)$$

Proof of Lemma 4: Because A extends to a continuous function on $(R \cup -\infty)^n \times \Delta$, it suffices to prove that it satisfies properties (1) and (2) of Definition 2. To prove property (1) note that

$$L(\gamma \circ q) - L(q) = \gamma \cdot \tilde{p}[f] - \log \sum_{\omega} q(\omega) e^{\gamma \cdot f(\omega)} \quad (10)$$

$$\geq \gamma \cdot \tilde{p}[f] + 1 - \sum_{\omega} q(\omega) e^{\gamma \cdot f(\omega)} \quad (11)$$

$$\geq \gamma \cdot \tilde{p}[f] + 1 - \sum_{\omega} q(\omega) \sum_i f(i|\omega) e^{\gamma_i f_{\#}(\omega)} \quad (12)$$

$$= A(\gamma, q). \quad (13)$$

Equality (10) is a simple calculation. Inequality (11) follows from inequality (9). Inequality (12) follows from the definition of $f_{\#}$ and Jensen's inequality (8). Property (2) of Definition 2 is straightforward to verify. \square

Proposition 3 follows immediately from the above lemma and the extended Proposition 5. Indeed, it is easy to check that $\gamma^{(k)}$ defined in Proposition 3 achieves the maximum of $A(\gamma, q^{(k)})$, so that it satisfies the condition of Proposition 5 in Appendix B.

D. Monte Carlo methods

The Improved Iterative Scaling algorithm described in the previous section is well-suited to numerical techniques since all of the features take non-negative values. In each iteration of this algorithm it is necessary to solve a polynomial equation for each feature f_i . That is, we can express equation 5 in the form

$$\sum_{m=0}^M a_{m,i}^{(k)} \beta_i^m = 0$$

where M is the largest value of $f_{\#}(\omega) = \sum_i f_i(\omega)$ and

$$a_{m,i}^{(k)} = \begin{cases} \sum_{\omega} q^{(k)}(\omega) f_i(\omega) \delta(m, f_{\#}(\omega)) & m > 0 \\ -\tilde{p}[f_i] & m = 0 \end{cases} \quad (14)$$

where $q^{(k)}$ is the field for the k -th iteration and $\beta_i = e^{\gamma_i^{(k)}}$. This equation has no solution precisely when $a_{m,i}^{(k)} = 0$ for $m > 0$. Otherwise, it can be efficiently solved using Newton's method since all of the coefficients $a_{m,i}^{(k)}$, $m > 0$, are non-negative. When Monte Carlo methods are to be used because the configuration space Ω is large, the coefficients $a_{m,i}^{(k)}$ can be simultaneously estimated for all i and m by generating a single set of samples from the distribution $q^{(k)}$.

V. APPLICATION: WORD MORPHOLOGY

Word clustering algorithms are useful for many natural language processing tasks. One such algorithm [6], called mutual information clustering, is based upon the construction of simple bigram language models using the maximum likelihood criterion. The algorithm gives a hierarchical binary classification of words that has been used for a variety of purposes, including the construction of decision tree language and parsing models, and sense disambiguation for machine translation [7].

A fundamental shortcoming of the mutual information word clustering algorithm given in [6] is that it takes as fundamental the word spellings themselves. This increases the severity of the problem of small counts that is present in virtually every statistical learning algorithm. For example, the word "Hamiltonianism" appears only once in the 365,893,263-word corpus used to collect bigrams for the clustering experiments described in [6]. Clearly this is insufficient evidence on which to base a statistical clustering decision. The basic motivation behind the feature-based approach is that by querying features of spellings, a clustering algorithm could notice that such a word begins with a capital letter, ends in "ism" or contains "ian," and profit from how these features are used for other words in similar contexts.

In this section we describe how we applied the random field induction algorithm to discover morphological features of words, and we present sample results. This application demonstrates how our technique gradually sharpens the probability mass from the enormous set of all possible configurations, in this case ASCII strings, onto a set of configurations that is increasingly similar to those in the training sample. It achieves this by introducing both "positive" features which many of the training samples exhibit, as well as "negative" features which do not appear in the sample, or appear only rarely. A description of how the resulting features

were used to improve mutual information clustering is given in [20], and is beyond the scope of the present paper; we refer the reader to [6], [20] for a more detailed treatment of this topic.

In Section 5.1 we formulate the problem in terms of the notation and results of Sections 2, 3, and 4. In Section 5.2 we describe how the field induction algorithm is actually carried out in this application. In Section 5.3 we explain the results of the induction algorithm by presenting a series of examples.

A. Problem formulation

To discover features of spellings we take as configuration space the set of all strings $\Omega = \mathcal{A}^*$ in the ASCII alphabet \mathcal{A} . We construct a probability distribution $p(\omega)$ on Ω by first predicting the length $|\omega|$, and then predicting the actual spelling; thus, $p(\omega) = p_l(|\omega|)p_s(\omega || \omega|)$ where p_l is the length distribution and p_s is the spelling distribution. We take the length distribution as given. We model the spelling distribution $p_s(\cdot || l)$ over strings of length l as a random field. Let Ω_l be the configuration space of all ASCII strings of length l . Then $|\Omega_l| = O(10^{2l})$ since each ω_i is an ASCII character.

To reduce the number of parameters, we tie features, as described in Section 2.1, so that a feature has the same weight independent of where it appears in the string. Because of this it is natural to view the graph underlying Ω_l as a regular l -gon. The group of automorphisms of this graph is the set of all rotations, and the resulting field is homogeneous as defined in Section 2.

Not only is each field p_s homogeneous, but in addition, we tie features across fields for different values of l . Thus, the weight λ_f of a feature is independent of l . To introduce a dependence on the length, as well as on whether or not a feature applies at the beginning or end of a string, we adopt the following artificial construction. We take as the graph of Ω_l an $(l+1)$ -gon rather than an l -gon, and label a distinguished vertex by the length, keeping this label held fixed.

To complete the description of the fields that are induced, we need to specify the set of atomic features. The atomic features that we allow fall into three types. The first type is the class of features of the form

$$f_{v,c}(\omega) = \begin{cases} 1 & \text{if } \omega_v = c \\ 0 & \text{otherwise.} \end{cases}$$

where c is any ASCII character, and v denotes an arbitrary character position in the string. The second type of atomic features involve the special vertex $\langle l \rangle$ that carries the length of the string. These are the features

$$\begin{aligned} f_{v,l}(\omega) &= \begin{cases} 1 & \text{if } \omega_v = \langle l \rangle \\ 0 & \text{otherwise} \end{cases} \\ f_{v,\langle l \rangle}(\omega) &= \begin{cases} 1 & \text{if } \omega_v = \langle l \rangle \text{ for some } l \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

The atomic feature $f_{v,\langle l \rangle}$ introduces a dependence on whether a string of characters lies at the beginning or end of the string, and the atomic features $f_{v,l}$ introduce a dependence on the length of the string. To tie together the length dependence for long strings, we also introduce an atomic feature $f_{v,7+}$ for strings of length 7 or greater.

The final type of atomic feature asks whether a character lies in one of four sets, $[a-z]$, $[A-Z]$, $[0-9]$, $[@-\&]$, denoting

arbitrary lowercase letters, uppercase letters, digits, and punctuation. For example, the atomic feature

$$f_{v,[a-z]}(\omega) = \begin{cases} 1 & \text{if } \omega_v \in [a-z] \\ 0 & \text{otherwise} \end{cases}$$

tests whether or not a character is lowercase.

To illustrate the notation that we use, let us suppose that the following features are active for a field: “ends in *ism*,” “a string of at least 7 characters beginning with a capital letter” and “contains *ian*.” Then the probability of the word “Hamiltonianism” would be given as

$$p_l(14) p_s(\text{Hamiltonianism} | |\omega| = 14) = p_l(14) \frac{1}{Z_{14}} e^{\lambda_{7+} \langle [A-Z] \rangle + \lambda_{ian} + \lambda_{ism}}.$$

Here the λ 's are the parameters of the appropriate features, and we use the characters \langle and \rangle to denote the beginning and ending of a string (more common regular expression notation would be \wedge and $\$$). The notation $7+ \langle [A-Z] \rangle$ thus means “a string of at least 7 characters that begins with a capital letter,” corresponding to the feature

$$f_{u,7+} f_{v,[A-Z]},$$

where u and v are *adjacent* positions in the string, recalling from Definition 2.1 that we require the support of a feature to be a connected subgraph. Similarly, $ism \rangle$ means “ends in *-ism*” and corresponds to the feature

$$f_{u,i} f_{v,s} f_{w,m} f_{x,\langle \rangle}$$

where u, v, w, x are adjacent positions in the string and ian means “contains *ian*,” corresponding to the feature

$$f_{u,i} f_{v,a} f_{w,n}.$$

B. Description of the algorithm

We begin the random field induction algorithm with a model that assigns uniform probability to all strings. We then incrementally add features to a random field model in order to minimize the Kullback-Leibler divergence between the field and the unigram distribution of the vocabulary obtained from a training corpus. The length distribution is taken according to the lengths of words in the empirical distribution of the training data. The improvement to the model made by a candidate feature is evaluated by the reduction in relative entropy, with respect to the unigram distribution, that adding the new feature yields, keeping the other parameters of the model fixed. Our learning algorithm incrementally constructs a random field to describe those features of spellings that are most informative.

At each stage in the induction algorithm, a set of candidate features is constructed. Because the fields are homogeneous, the set of candidate features can be viewed as follows. Each active feature can be expressed in the form

$$f_s(\omega) = \begin{cases} 1 & \text{substring } s \text{ appears in } \omega \\ 0 & \text{otherwise} \end{cases}$$

where s is a string in the extended alphabet \mathcal{A} of ASCII characters together with the macros $[a-z]$, $[A-Z]$, $[0-9]$, $[@-\&]$, and

the length labels $\langle 1 \rangle$ and $\langle \cdot \rangle$. If $\{f_s\}_{s \in S}$ is the set of active features, (including $s = \epsilon$, the empty string) using this representation, then the set of candidate features is precisely the set $\{f_{a \cdot s}, f_{s \cdot a}\}_{a \in A, s \in S}$, where $a \cdot s$ denotes concatenation of strings. As required by Definition 2, each such candidate increases the support of an active feature by a single adjacent vertex.

Since the model assigns probability to arbitrary word strings, the partition function Z_l can be computed exactly for only the smallest string lengths l . We therefore compute feature expectations using a random sampling algorithm. Specifically, we use the Gibbs sampler to generate 10,000 spellings of random lengths. When computing the gain $G_q(g)$ of a candidate feature, we use these spellings to estimate the probability g_k that the candidate feature g occurs k times in a spelling (see equation (4)—for example, the feature $f_{v, [a-z]}$ occurs two times in the string The), and then solve for the corresponding β using Newton’s method for *each* candidate feature. It should be emphasized that only a single set of random spellings needs to be generated; the same set can be used to estimate g_k for each candidate g . After adding the best candidate to the field, all of the feature weights are readjusted using the Improved Iterative Scaling algorithm. To carry out this algorithm, random spellings are again generated, this time incorporating the new feature, yielding Monte Carlo estimates of the coefficients $a_{m,i}^{(k)}$. Recall that $a_{m,i}^{(k)}$ is the expected number of times that feature i appears (under the substring representation for homogeneous features) in a string for which there is a total of m active features (see equation 14)). Given estimates for these coefficients, Newton’s method is again used to solve equation (14), to complete a single iteration of the iterative scaling algorithm. After convergence of the Kullback-Leibler divergence, the inductive step is complete, and a new set of candidate features is considered.

C. Sample results

We began with a uniform field, that is, a field with no features at all. For this field, all ASCII strings of a given length are equally likely, and the lengths are drawn from a fixed distribution. Here is a sample of strings drawn from this distribution:

```

~, mo, _!ZP*@, m/TLL, ks;cm_3, *LQdR, D, aW{,
5&TL|4, tc, ?!@, sNeiO+, wHo8zBr", pQLV, m, H!&,
h9, #Os, :, Ky}FM?, LW, ", 8}, 89Lj, -P, A, !, H, ` ,
Y^:Du:, lxCl, 1!'J#F*u., w=idHnM), ~, 2, 2leW2,
I,bw~tk1, 3", ], ], b, +JEmj6, +E*, \qjqe"-7f, |a12,
T, ~(sOc1+2ADe, &, \p9oH, i;, $6, q}O+[ , xEv, #U,
O)[83COF, =|B|7%cR, Mqq, ?!mv, n=7G, $i9GAJ D, 5,
, =, +u6@I9:, +, =D, 2E#vz@3-, ~nu;.+s, 3xJ, GDWeqL,
R, 3R, !7v, FX,@y, 4p.cY2hU, ~
    
```

It comes as no surprise that the first feature the induction algorithm chooses is $[a-z]$; it simply observes that characters should be lowercase. The maximum likelihood (maximum entropy) weight for this feature is $\beta = e^\lambda \approx 6.99$. This means that a string with a lowercase letter in some position is about 7 times more likely than the same string without a lowercase letter in that position.

When we now draw strings from the new distribution (using annealing to concentrate the distribution on the more probable strings), we obtain spellings that are primarily made up of lowercase letters, but that certainly do not resemble English words:

```

m, r, xevo, ijjiir, b, to, jz, gsr, wq, vf, x, ga,
msmGH, pcp, d, oziVlal, hzagh, yzop, io, advzmxnv,
ijv.bolft, x, emx, kayerf, mlj, rawzyb, jp, ag,
ctdnbbg, wgdw, t, kguv, cy, spxcq, uzflbbf,
dxtkkn, cxwx, jpd, ztzh, lv, zhpkvnu, l; r, qee,
nynrx, atze4n, ik, se, w, lrh, hp+, yrqyka'h,
zcnogtcnx, igcump, zjcjs, lqpWiqu, cefmfhc, o, lb,
fdcY, tzby, yopxmvk, by, fz,, t, govycm,
ijyiduwfzo, 6xr, duh, ejv, pk, pjw, l, fl, w
    
```

In the following table we show the first 10 features that the algorithm induced, together with their associated parameters. Several things are worth noticing. The second feature chosen was $[a-z][a-z]$, which denotes adjacent lowercase characters. The third feature added was the letter e, which is the most common letter. The weight for this feature is $\beta = e^\lambda = 3.47$. The next feature introduces the first dependence on the length of the string: $[a-z]>1$ denotes the feature “a one character word ending with a lowercase letter.” Notice that this feature has a small weight of 0.04, corresponding to our intuition that such words are uncommon. Similarly, the features z, q, j, and x are uncommon, and thus receive small weights. The appearance of the feature * is explained by the fact that the vocabulary for our corpus is restricted to the most frequent 100,000 spellings, and all other words receive the “unknown word” spelling ***, which is rather frequent. (The “end-of-sentence” marker, which makes its appearance later, is given the spelling |.)

feature	$[a-z]$	$[a-z][a-z]$	e	$[a-z]>1$	t
β	6.64	6.07	3.47	0.04	2.75
feature	*	z	q	j	x
β	17.25	0.02	0.03	0.02	0.06

Shown below are spellings obtained by Gibbs sampling from the resulting collection of fields.

```

frk, et, egeit, edet, eutdmeeet, ppge, A, dtgd,
falawe, etci, eese, ye, epemtbn, tegoeed, ee, *mp,
temou, enrteunt, ore, erveelew, heyu, rht, *,
lkaeu, lutoee, tee, mmo, eobwtit, weethtw, 7, ee,
teet, gre, /, *, eeeteetue, hgtte, om, he, *,
stmenu, ec, ter, eedgtue, iu, ec, reett, *,
ivtmeeee, vt, eets, tidpt, lttv, *, etttvti, ecte,
X, see, *, pi, rlet, tt, *, eot, leef, ke, *, *,
tet, iwteeiwbeie, yeee, et, etf, *, ov
    
```

After inducing 100 features, the model finally begins to be concentrated on spellings that resemble actual words to some extent, particularly for short strings. At this point the algorithm has discovered, for example, that the is a very common 3-letter word, that many words end in ed, and that long words often end in ion. A sample of 10 of the first 100 features induced, with their appropriate weights is shown in the table below.

.	,>1	3<the	tion	4<th	y>	ed>	ion>7+	ent	7+<c
22.36	31.30	11.05	5.89	4.78	5.35	4.20	4.83	5.17	5.37

```

thed, and, thed, toftion, |, ieention, cention, |,
ceetion, ant, is, seieeet, cinention, and, ..,
tlned, uointe, feredten, iined, sonention,
inathed, other, the, id, and, .., of, is, of, of, ..,
lcers, .., ceecion, .., roferented, |, ioner, .., |,
the, the, the, centention, ionent, asers, ..,
ctention, |, of, thed, of, uentie, of, and, ttentt,
in, rerey, and, |, sotth, cheent, is, and, of,
thed, rontion, that, seoftr
    
```

A sample of the first 1000 features induced is shown in the table below, together with randomly generated spellings. Notice, for example, that the feature $[0-9][0-9]$ appears with a surprisingly high weight of 9382.93. This is due to the fact that if a string contains one digit, then it's very likely to contain two digits. But since digits are relatively rare in general, the feature $[0-9]$ is assigned a small weight of 0.038. Also, according to the model, a lowercase letter followed by an uppercase letter is rare.

s>	<re	ght>	3<[A-Z]	ly>	al>7+	ing>
7.25	4.05	3.71	2.27	5.30	94.19	16.18
[a-z][A-Z]	't>	ed>7+	er>7+	ity	ent>7+	[0-9][0-9]
0.003	138.56	12.58	8.11	4.34	6.12	9382.93
qu	ex	ae	ment	ies	<wh	ate
526.42	5.265	0.001	10.83	4.37	5.26	9.79

was, reaser, in, there, to, will, ,, was, by, homes, thing, be, reloverated, ther, which, consists, at, fores, anditing, with, Mr., proveral, the, ,, ***, on't, prolling, prothere, ,, mento, at, yaou, l, chestraing, for, have, to, intrally, of, gut, ,, best, compers, ***, cluseliment, uster, of, is, deveral, this, thise, of, offect, inatever, thifer, constrandred, stater, vill, in, thase, in, youse, menttering, and, ,, of, in, verate, of, to

Finally, we visit the state of the model after inducing 1500 features to describe words. At this point the model is making more refined judgements regarding what is to be considered a word and what is not. The appearance of the features $\{ \} >$ and $\{ @- \& \} \{$, is explained by the fact that in preparing our corpus, certain characters were assigned special "macro" strings. For example, the punctuation characters \$, _, %, and & are represented in our corpus as $\backslash \$ \{ \}$, $\backslash _ \{ \}$, $\backslash \% \{ \}$, and $\backslash \& \{ \}$. As the following sampled spellings demonstrate, the model has at this point recognized the existence of macros, but has not yet discerned their proper use.

7+<inte	prov	<der	<wh	19	ons>7+	ugh	ic>
4.23	5.08	0.03	2.05	2.59	4.49	5.84	7.76
sys	ally	7+<con	ide	nal	{ } >	qui	\{ @- \& \} \{
4.78	6.10	5.25	4.39	2.91	120.56	18.18	913.22
iz	IB	<inc	<im	iong	\$	ive>7+	<un
10.73	10.85	4.91	5.01	0.001	16.49	2.83	9.08

the, you, to, by, conthing, the, ,, not, have, devened, been, of, |, F., ,, in, have, -, ,, intering, ***, ation, said, prouned, ***, suparthere, in, mentter, prement, intever, you, ,, and, B., gover, producuts, alase, not, conting, comment, but, |, that, of, is, are, by, from, here, incements, contive, ,, evined, agents, and, be, ', thent, distements, all, --, has, will, said, resting, had, this, was, intevent, IBM, whree, acalinate, herved, are, ***, O., |, 1980, but, will, ***, is, ,, to, becoment, ,, with, recall, has, |, nother, ments, was, the, to, of, stounicallity, with, camanfined, in, this, intations, it, conanament, out, they, you

While clearly the model still has much to learn, it has at this point compiled a significant collection of morphological observations, and has traveled a long way toward its goal of statistically characterizing English spellings.

VI. EXTENSIONS AND RELATIONS TO OTHER APPROACHES

In this section we briefly discuss some relations between our incremental feature induction algorithm for random fields and other statistical learning paradigms. We also present some possible extensions and improvements of our method.

A. Conditional exponential models

Almost all of what we have presented here carries over to the more general setting of conditional exponential models, including the Improved Iterative Scaling algorithm. For general conditional distributions $p(y|x)$ there may be no underlying random field, but with features defined as binary functions $f(x,y)$, the same general approach is applicable. The feature induction method for conditional exponential models is demonstrated for several problems in statistical machine translation in [3], where it is presented in terms of the principle of maximum entropy.

B. Decision trees

Our feature induction paradigm also bears some resemblance to various methods for growing classification and regression trees. Like decision trees, our method builds a top-down classification that refines features. However, decision trees correspond to constructing features that have disjoint support.

To explain, recall that a decision tree determines a partition π of a context random variable $X \in \mathcal{X}$ in order to predict the actual class of the context, represented by a random variable $Y \in \mathcal{Y}$. Each leaf in the tree corresponds to a sequence of binary features

$$f_l, f_{l \uparrow}, f_{l \uparrow \uparrow}, \dots, f_{\text{root}}$$

where $n \uparrow$ denotes the parent of node n , each feature f_n is a question which splits \mathcal{X} , and where each f_n is the negation $\neg f_n$ of the question asked at its sibling node. The distribution assigned to a leaf l is simply the empirical distribution on \mathcal{Y} determined by the training samples $(x,y) \in \mathcal{X} \times \mathcal{Y}$ for which $\pi(x) = l$. Each leaf l is characterized by the conjunction of these features, and different leaves correspond to conjunctions with disjoint support. In contrast, our feature induction algorithm generally results in features that have overlapping support. The criterion of evaluating questions in terms of the amount by which they reduce the conditional entropy of Y corresponds to our criterion of maximizing the reduction in Kullback-Leibler divergence, $G_q(g)$, over all candidate features g for a field q .

By modifying our induction algorithm in the following way, we obtain an algorithm closely related to standard methods for growing binary decision trees. Instead of considering the 1-parameter family of fields $q_{\lambda,g}$ to determine the best candidate $g = a \wedge f$, we consider the 2-parameter family of fields given by

$$q_{\lambda,\lambda',f}(y|x) = \frac{1}{Z_{\lambda,\lambda',f}(x)} e^{\lambda a(x,y) \wedge f(x,y) + \lambda' (\neg a)(x,y) \wedge f(x,y)}$$

Since the features $a \wedge f$ and $(\neg a) \wedge f$ have disjoint support, the improvement obtained by adding both of them is given by $G_q(a \wedge f) + G_q((\neg a) \wedge f)$. In general, the resulting distribution is not absolutely continuous with respect to the empirical distribution. If the random variable Y can take on M values y_1, \dots, y_M , then the standard decision tree algorithm is obtained if at the n -th stage

we add the $2M$ (disjoint) features $f_n(x) \wedge \delta(y_i, y)$, $\neg f_n(x) \wedge \delta(y_i, y)$, for $i = 1, \dots, M$. Maximum likelihood training of the parameters of these features recovers the empirical distribution of the data at node n .

C. Extensions

As mentioned in Section 1, our approach differs from the most common applications of statistical techniques in computer vision, since a typical application of our method involves the estimation of thousands of free parameters. Yet the induction technique may not scale well to large 2-dimensional image problems. One potential difficulty is that the degree of the polynomials in the Improved Iterative Scaling algorithm could be quite large, and it could be difficult to obtain reliable estimates of the coefficients since Monte Carlo sampling might not exhibit sufficiently many instances of the desired features. The extent to which this is a significant problem is primarily an empirical issue, dependent on the particular domain to which the method is applied.

The random field induction method presented in this paper is not definitive; there are many possible variations on the basic theme, which is to incrementally construct an increasingly detailed exponential model to approximate the reference distribution \tilde{p} . Because the basic technique is based on a greedy algorithm, there are of course many ways for improving the search for a good set of features. The algorithm presented in Section 2 is in some respects the most simple possible within the general framework. But it is also computationally intensive. A natural modification would be to add several of the top candidates at each stage. While this should increase the overall speed of the induction algorithm, it would also potentially result in more redundancy among the features, since the top candidates could be correlated. Another modification of the algorithm would be to add only the best candidate at each step, but then to carry out parameter estimation only after several new features had been added to the field. It would also be natural to establish a more Bayesian framework in which a prior distribution on features and parameters is incorporated. This could enable a principled approach for deciding when the feature induction is complete. While there is a natural class of conjugate priors for the class of exponential models that we use [14], the problem of incorporating prior knowledge about the set of candidate features is more challenging.

APPENDIX

I. DUALITY

In this Appendix we prove Proposition 4 restated here.

Proposition 4: Suppose that $\tilde{p} \ll q_0$. Then there exists a unique q_* $\in \Delta$ satisfying

- (1) $q_* \in \mathcal{P} \cap \tilde{\mathcal{Q}}$
- (2) $D(p \| q) = D(p \| q_*) + D(q_* \| q)$ for any $p \in \mathcal{P}$ and $q \in \tilde{\mathcal{Q}}$
- (3) $q_* = \arg \min_{q \in \tilde{\mathcal{Q}}} D(\tilde{p} \| q)$
- (4) $q_* = \arg \min_{p \in \mathcal{P}} D(p \| q_0)$.

Moreover, any of these four properties determines q_* uniquely.

Our proof of the proposition will use a few lemmas. The first two lemmas we state without proof.

Lemma 1:

- (1) $D(p \| q)$ is a non-negative, extended real-valued function on $\Delta \times \Delta$.
- (2) $D(p \| q) = 0$ if and only if $p = q$.
- (3) $D(p \| q)$ is strictly convex in p and q separately.
- (4) $D(p \| q)$ is C^1 in q .

Lemma 2:

- (1) The map $(\gamma, p) \mapsto \gamma \circ p$ is smooth in $(\gamma, p) \in \mathbf{R}^n \times \Delta$.
- (2) The derivative of $D(p \| \lambda \circ q)$ with respect to λ is

$$\left. \frac{d}{dt} \right|_{t=0} D(p \| (t\lambda) \circ q) = \lambda \cdot (p[f] - q[f]).$$

Lemma 3: If $\tilde{p} \ll q_0$ then $\mathcal{P} \cap \tilde{\mathcal{Q}}$ is nonempty.

Proof: Define q_* by property (3) of Proposition 4; that is, $q_* = \arg \min_{q \in \tilde{\mathcal{Q}}} D(\tilde{p} \| q)$. To see that this makes sense, note that since $\tilde{p} \ll q_0$, $D(\tilde{p} \| q)$ is not identically ∞ on $\tilde{\mathcal{Q}}$. Also, $D(p \| q)$ is continuous and strictly convex as a function of q . Thus, since $\tilde{\mathcal{Q}}$ is closed, $D(\tilde{p} \| q)$ attains its minimum at a unique point $q_* \in \tilde{\mathcal{Q}}$. We will show that q_* is also in \mathcal{P} . Since $\tilde{\mathcal{Q}}$ is closed under the action of \mathbf{R}^n , $\lambda \circ q_*$ is in $\tilde{\mathcal{Q}}$ for any λ . Thus by the definition of q_* , $\lambda = 0$ is a minimum of the function $\lambda \rightarrow D(\tilde{p} \| \lambda \circ q_*)$. Taking derivatives with respect to λ and using Lemma A.2 we conclude $q_*[f] = \tilde{p}[f]$. Thus $q_* \in \mathcal{P}$. \square

Lemma 4: If $q_* \in \mathcal{P} \cap \tilde{\mathcal{Q}}$ then for any $p \in \mathcal{P}$ and $q \in \tilde{\mathcal{Q}}$

$$D(p \| q) = D(p \| q_*) + D(q_* \| q).$$

Proof: A straightforward calculation shows that

$$\begin{aligned} D(p_1 \| q_1) - D(p_1 \| q_2) - D(p_2 \| q_1) + D(p_2 \| q_2) \\ = \lambda \cdot (p_1[f] - p_2[f]) \end{aligned}$$

for any $p_1, p_2, q_1, q_2 \in \Delta$ with $q_2 = \lambda \circ q_1$. It follows from this identity and the continuity of D that

$$D(p_1 \| q_1) - D(p_1 \| q_2) - D(p_2 \| q_1) + D(p_2 \| q_2) = 0$$

if $p_1, p_2 \in \mathcal{P}$ and $q_1, q_2 \in \tilde{\mathcal{Q}}$. The lemma follows by taking $p_1 = p_2 = q_*$. \square

Proof of Proposition 4: Choose q_* to be any point in $\mathcal{P} \cap \tilde{\mathcal{Q}}$. Such a q_* exists by Lemma A.3. It satisfies property (1) by definition, and it satisfies property (2) by Lemma A.4. As a consequence of property (2), it also satisfies properties (3) and (4). To check property (3), for instance, note that if q is any point in $\tilde{\mathcal{Q}}$, then $D(\tilde{p} \| q) = D(\tilde{p} \| q_*) + D(q_* \| q) \geq D(\tilde{p} \| q_*)$.

It remains to prove that each of the four properties (1)–(4) determines q_* uniquely. In other words, we need to show that if m is any point in Δ satisfying any of the four properties (1)–(4), then $m = q_*$. Suppose that m satisfies property (1). Then by property (2) for q_* with $p = q = m$, $D(m \| m) = D(m \| q_*) + D(q_* \| m)$. Since $D(m \| m) = 0$, it follows that $D(m, q_*) = 0$ so $m = q_*$. If m satisfies property (2), then the same argument with q_* and m reversed again proves that $m = q_*$. Suppose that m satisfies property (3). Then

$$D(\tilde{p} \| q_*) \geq D(\tilde{p} \| m) = D(\tilde{p} \| q_*) + D(q_* \| m)$$

where the second equality follows from property (2) for q_* . Thus $D(q_* \| m) \leq 0$ so $m = q_*$. If m satisfies property (4), then a similar proof shows that once again $m = q_*$. \square

II. DEALING WITH $-\infty$

In this Appendix we prove an extension of Proposition 5 that allows the components of γ to equal $-\infty$. For this extension, we assume that all of the components of the feature function f are non-negative: $f_i(\omega) \geq 0$ for all i and all ω . This can be assumed with no loss of generality since we can replace f_i by $f_i - \min_{\omega} f_i(\omega)$ if necessary.

Let $R \cup -\infty$ denote the partially extended real numbers with the usual topology. The operations of addition and exponentiation extend continuously to $R \cup -\infty$. Let \mathcal{S} be the open subset of $(R \cup -\infty)^n \times \Delta$ defined by

$$\mathcal{S} = \{ (\gamma, q) : q(\omega) e^{\gamma \cdot f(\omega)} > 0 \text{ for some } \omega \}$$

Observe that $R^n \times \Delta$ is a dense subset of \mathcal{S} . The map $(\gamma, q) \mapsto \gamma \circ p$, which up to this point we defined only for finite γ , extends uniquely to a continuous map from all of \mathcal{S} to Δ . (The condition on $(\gamma, q) \in \mathcal{S}$ ensures that the normalization in the definition of $\gamma \circ p$ is well defined, even if γ is not finite.)

Definition 3: We call a function $A : \mathcal{S} \rightarrow R \cup -\infty$ an *extended auxiliary function* for L if when restricted to $R^n \times \Delta$ it is an ordinary auxiliary function in the sense of Definition 2, and if, in addition, it satisfies property (1) of Definition 2 for any $(q, \gamma) \in \mathcal{S}$, even if γ is not finite.

Note that if an ordinary auxiliary function extends to a continuous function on \mathcal{S} , then the extension is an extended auxiliary function.

We have the following extension of Proposition 5:

Proposition 5: Suppose the feature function f satisfies the non-negativity condition 7 and suppose A is an extended auxiliary function for L . Then the conclusion of Proposition 5 continues to hold if the condition on $\gamma^{(k)}$ is replaced by: $(\gamma^{(k)}, q^{(k)}) \in \mathcal{S}$ and $A(\gamma^{(k)}, q^{(k)}) \geq A(\gamma, q^{(k)})$ for any $(\gamma, q^{(k)}) \in \mathcal{S}$.

Proof: Lemma 1 is valid under the altered condition on $\gamma^{(k)}$ since $A(\gamma, q)$ satisfies property (1) of Definition 2 for all $(\gamma, q) \in \mathcal{S}$. As a consequence, Lemma 2 also is valid, and the proof of Proposition 5 goes through without change. \square

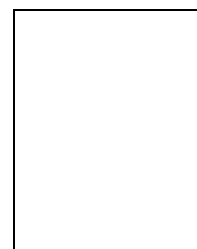
III. ACKNOWLEDGEMENTS

Part of the research presented in this paper was carried out while the authors were with the IBM Thomas J. Watson Research Center in Yorktown Heights, New York. Stephen Della Pietra and Vincent Della Pietra's work was partially supported by ARPA under grant N00014-91-C-0135. John Lafferty's work was partially supported by NSF and ARPA under grants IRI-9314969 and N00014-92-C-0189.

REFERENCES

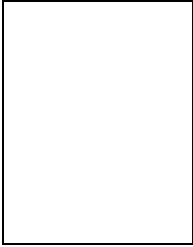
- [1] M. Almeida and B. Gidas, "A variational method for estimating the parameters of MRF from complete or incomplete data," *The Annals of Applied Probability*, **3**, No. 1, 103–136, 1993.
- [2] N. Balram and J. Moura, "Noncausal Gauss Markov random fields: Parameter structure and estimation," *IEEE Transactions on Information Theory* **39**, No. 4, 1333–1343, July, 1993.

- [3] A. Berger, V. Della Pietra, and S. Della Pietra, "A maximum entropy approach to natural language processing," *Computational Linguistics*, **22**, No. 1, 39–71, 1996.
- [4] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Wadsworth, Belmont, 1984.
- [5] D. Brown, "A note on approximations to discrete probability distributions," *Information and Control* **2**, 386–392 (1959).
- [6] P. Brown, V. Della Pietra, P. de Souza, J. Lai, and R. Mercer, "Class-based n -gram models of natural language," *Computational Linguistics* **18**, No. 4, 467–479, 1992.
- [7] P. F. Brown, J. Cocke, V. Della-Pietra, S. Della-Pietra, J. D. Lafferty, R. L. Mercer, and P. S. Roossin, "A statistical approach to machine translation," *Computational Linguistics*, **16**(2):79–85, 1990.
- [8] B. Chalmoud, "An iterative Gibbsian technique for reconstruction of m -ary images," *Pattern Recognition*, **22** No. 6, 747–761, 1989.
- [9] I. Csiszár, "I-Divergence geometry of probability distributions and minimization problems," *The Annals of Probability*, **3**, No. 1, 146–158, 1975.
- [10] I. Csiszár, "A geometric interpretation of Darroch and Ratcliff's generalized iterative scaling," *The Annals of Statistics*, **17**, No. 3, 1409–1413, 1989.
- [11] I. Csiszár and G. Tusnády, "Information geometry and alternating minimization procedures," *Statistics & Decisions*, Supplement Issue, **1**, 205–237, 1984.
- [12] J. Darroch and D. Ratcliff, "Generalized iterative scaling for log-linear models," *Ann. Math. Statist.* **43**, 1470–1480, 1972.
- [13] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society* **39**, No. B, 1–38, 1977.
- [14] P. Diaconis and D. Ylvisaker, "Conjugate priors for exponential families," *Ann. Statist.* **7**, 269–281, 1979.
- [15] P. Ferrari, A. Frigessi and R. Schonmann, "Convergence of some partially parallel Gibbs samplers with annealing," *The Annals of Applied Probability*, **3** No. 1, 137–152, 1993.
- [16] A. Frigessi, C. Hwang, and L. Younes, "Optimal spectral structure of reversible stochastic matrices, Monte Carlo methods and the simulation of Markov random fields," *The Annals of Applied Probability*, **2**, No. 3, 610–628, 1992.
- [17] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Machine Intell.* **6**, 721–741, 1984.
- [18] C. Geyer and E. Thomson, "Constrained Monte Carlo maximum likelihood for dependent data (with discussion)," *J. Royal Stat. Soc. B-54*, 657–699, 1992.
- [19] E. T. Jaynes, *Papers on Probability, Statistics, and Statistical Physics*, R. Rosenkrantz, ed., D. Reidel Publishing Co., Dordrecht–Holland, 1983.
- [20] J. Lafferty and R. Mercer, "Automatic word classification using features of spellings," *Proceedings of the 9th Annual Conference of the University of Waterloo Centre for the New OED and Text Research*, Oxford University Press, Oxford, England, 1993.
- [21] G. Potamianos and J. Goutsias, "Partition function estimation of Gibbs random field images using Monte Carlo simulations," *IEEE Transactions on Information Theory* **39**, No. 4, 1322–1331, July, 1993.
- [22] L. Younes, "Estimation and annealing for Gibbsian fields," *Ann. Inst. H. Poincaré Probab. Statist.* **24** No. 2, 269–294, 1988.



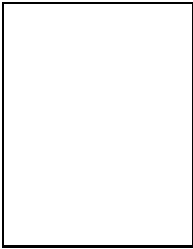
Stephen A. Della Pietra received his A.B. in Physics from Princeton University in 1981, and Ph.D. in Physics from Harvard University in 1987. From 1987 until 1988 he was a Post-Doctoral Fellow at the University of Texas at Austin. From 1988 until 1989, he was a Member of the Mathematics Department of the Institute for Advanced Study in Princeton, NJ. From 1989 until 1995 he was a Research Staff Member at the IBM Thomas J. Watson Research Center in Yorktown Heights and Hawthorne, NY, where he was project leader of the natural language understanding group.

His primary research at IBM was in machine translation and natural language understanding. Since 1995 he has been working on statistical methods for modeling the stock market at Renaissance Technologies in Stony Brook, NY.



Vincent J. Della Pietra received his A.B. in Mathematics from Princeton University in 1982, and Ph.D. in Mathematical Physics from Harvard University in 1988. From 1987 until 1988, he was a Member of the Mathematics Department of the Institute for Advanced Study in Princeton, NJ. From 1988 until 1995 he was a Research Staff Member at the IBM Thomas J. Watson Research Center in Yorktown Heights and Hawthorne, NY, where he was project leader of the machine translation group. His primary research at IBM was in machine translation and natural language understanding.

Since 1995 he has been working on statistical methods for modeling the stock market at Renaissance Technologies in Stony Brook, NY.



John D. Lafferty studied mathematics as an undergraduate at Middlebury College and the Massachusetts Institute of Technology, and received the Ph.D. degree in Mathematics from Princeton University in 1986, where he was a member of the Program in Applied and Computational Mathematics. From 1986 until 1987 he was an Assistant Professor and Benjamin Pierce Lecturer on Mathematics at Harvard University, and from 1987 until 1988 he was Visiting Assistant Professor at the Nankai Institute of Mathematics in Tianjin, China.

In 1989 he joined the Computer Sciences Department of the IBM Thomas J. Watson Research Center in Yorktown Heights, NY, as a Research Staff Member. Since 1994 he has been a member of the faculty of the Computer Science Department at Carnegie Mellon University, where his primary research interests include speech, language, information theory and statistical learning algorithms.

Estimators for Stochastic “Unification-Based” Grammars*

Mark Johnson **Stuart Geman** **Stephen Canon**
Cognitive and Linguistic Sciences Applied Mathematics Cognitive and Linguistic Sciences
Brown University Brown University Brown University

Zhiyi Chi **Stefan Riezler**
Dept. of Statistics Institut für Maschinelle Sprachverarbeitung
The University of Chicago Universität Stuttgart

Abstract

Log-linear models provide a statistically sound framework for Stochastic “Unification-Based” Grammars (SUBGs) and stochastic versions of other kinds of grammars. We describe two computationally-tractable ways of estimating the parameters of such grammars from a training corpus of syntactic analyses, and apply these to estimate a stochastic version of Lexical-Functional Grammar.

1 Introduction

Probabilistic methods have revolutionized computational linguistics. They can provide a systematic treatment of preferences in parsing. Given a suitable estimation procedure, stochastic models can be “tuned” to reflect the properties of a corpus. On the other hand, “Unification-Based” Grammars (UBGs) can express a variety of linguistically-important syntactic and semantic constraints. However, developing Stochastic “Unification-based” Grammars (SUBGs) has not proved as straightforward as might be hoped.

The simple “relative frequency” estimator for PCFGs yields the maximum likelihood parameter estimate, which is to say that it minimizes the Kulback-Liebler divergence between the training and estimated distributions. On the other hand, as Abney (1997) points out, the context-sensitive dependencies that “unification-based” constraints introduce render the relative frequency estimator suboptimal: in general it does not maximize the likelihood and it is inconsistent.

Abney (1997) proposes a Markov Random Field or log linear model for SUBGs, and the models described here are instances of Abney’s general framework. However, the Monte-Carlo parameter estimation procedure that Abney proposes seems to be computationally impractical for reasonable-sized grammars. Sections 3 and 4 describe two new estimation procedures which are computationally tractable. Section 5 describes an experiment with a small LFG corpus provided to us by Xerox PARC. The log linear framework and the estimation procedures are extremely general, and they apply directly to stochastic versions of HPSG and other theories of grammar.

2 Features in SUBGs

We follow the statistical literature in using the term *feature* to refer to the properties that parameters are associated with (we use the word “attribute” to refer to the attributes or features of a UBG’s feature structure). Let Ω be the set of all possible grammatical or well-formed analyses. Each feature f maps a syntactic analysis $\omega \in \Omega$ to a real value $f(\omega)$. The form of a syntactic analysis depends on the underlying linguistic theory. For example, for a PCFG ω would be parse tree, for a LFG ω would be a tuple consisting of (at least) a c-structure, an f-structure and a mapping from c-structure nodes to f-structure elements, and for a Chomskyian transformational grammar ω would be a derivation.

Log-linear models are models in which the log probability is a linear combination of feature values (plus a constant). PCFGs, Gibbs distributions, Maximum-Entropy distributions and Markov Random Fields are all examples of log-linear models. A log-linear model associates each feature f_j with a real-valued parameter θ_j .

* This research was supported by the National Science Foundation (SBR-9720368), the US Army Research Office (DAAH04-96-BAA5), and Office of Naval Research (N00014-97-1-0249).

A log-linear model with m features is one in which the likelihood $P(\omega)$ of an analysis ω is:

$$P_{\theta}(\omega) = \frac{1}{Z_{\theta}} e^{\sum_{j=1, \dots, m} \theta_j f_j(\omega)}$$

$$Z_{\theta} = \sum_{\omega' \in \Omega} e^{\sum_{j=1, \dots, m} \theta_j f_j(\omega')}$$

While the estimators described below make no assumptions about the range of the f_i , in the models considered here the value of each feature $f_i(\omega)$ is the number of times a particular structural arrangement or configuration occurs in the analysis ω , so $f_i(\omega)$ ranges over the natural numbers.

For example, the features of a PCFG are indexed by productions, i.e., the value $f_i(\omega)$ of feature f_i is the number of times the i th production is used in the derivation ω . This set of features induces a tree-structured dependency graph on the productions which is characteristic of Markov Branching Processes (Pearl, 1988; Frey, 1998). This tree structure has the important consequence that simple “relative-frequencies” yield maximum-likelihood estimates for the θ_i .

Extending a PCFG model by adding additional features not associated with productions will in general add additional dependencies, destroy the tree structure, and substantially complicate maximum likelihood estimation.

This is the situation for a SUBG, even if the features are production occurrences. The unification constraints create non-local dependencies among the productions and the dependency graph of a SUBG is usually not a tree. Consequently, maximum likelihood estimation is no longer a simple matter of computing relative frequencies. But the resulting estimation procedures (discussed in detail, shortly), albeit more complicated, have the virtue of applying to essentially arbitrary features—of the production or non-production type. That is, since estimators capable of finding maximum-likelihood parameter estimates for production features in a SUBG will also find maximum-likelihood estimates for non-production features, there is no motivation for restricting features to be of the production type.

Linguistically there is no particular reason for assuming that productions are the best features to use in a stochastic language model.

For example, the adjunct attachment ambiguity in (1) results in alternative syntactic structures which use the same productions the same number of times in each derivation, so a model with only production features would necessarily assign them the same likelihood. Thus models that use production features alone predict that there should not be a systematic preference for one of these analyses over the other, contrary to standard psycholinguistic results.

1.a Bill thought Hillary [_{VP}[_{VP} left] yesterday]

1.b Bill [_{VP}[_{VP} thought Hillary left] yesterday]

There are many different ways of choosing features for a SUBG, and each of these choices makes an empirical claim about possible distributions of sentences. Specifying the features of a SUBG is as much an empirical matter as specifying the grammar itself. For any given UBG there are a large (usually infinite) number of SUBGs that can be constructed from it, differing only in the features that each SUBG uses.

In addition to production features, the stochastic LFG models evaluated below used the following kinds of features, guided by the principles proposed by Hobbs and Bear (1995). Adjunct and argument features indicate adjunct and argument attachment respectively, and permit the model to capture a general argument attachment preference. In addition, there are specialized adjunct and argument features corresponding to each grammatical function used in LFG (e.g., SUBJ, OBJ, COMP, XCOMP, ADJUNCT, etc.). There are features indicating both high and low attachment (determined by the complexity of the phrase being attached to). Another feature indicates non-right-branching nonterminal nodes. There is a feature for non-parallel coordinate structures (where parallelism is measured in constituent structure terms). Each f-structure attribute-atomic value pair which appears in any feature structure is also used as a feature. We also use a number of features identifying syntactic structures that seem particularly important in these corpora, such as a feature identifying NPs that are dates (it seems that date interpretations of NPs are preferred). We would have liked to have included features concerning specific lexical items (to capture head-to-head dependencies), but we felt that our corpora were so small

that the associated parameters could not be accurately estimated.

3 A pseudo-likelihood estimator for log linear models

Suppose $\tilde{\omega} = \omega_1, \dots, \omega_n$ is a training corpus of n syntactic analyses. Letting $f_j(\tilde{\omega}) = \sum_{i=1, \dots, n} f_j(\omega_i)$, the log likelihood of the corpus $\tilde{\omega}$ and its derivatives are:

$$\log L_\theta(\tilde{\omega}) = \sum_{j=1, \dots, m} \theta_j f_j(\tilde{\omega}) - n \log Z_\theta \quad (2)$$

$$\frac{\partial \log L_\theta(\tilde{\omega})}{\partial \theta_j} = f_j(\tilde{\omega}) - n E_\theta(f_j) \quad (3)$$

where $E_\theta(f_j)$ is the expected value of f_j under the distribution determined by the parameters θ . The maximum-likelihood estimates are the θ which maximize $\log L_\theta(\tilde{\omega})$. The chief difficulty in finding the maximum-likelihood estimates is calculating $E_\theta(f_j)$, which involves summing over the space of well-formed syntactic structures Ω . There seems to be no analytic or efficient numerical way of doing this for a realistic SUBG.

Abney (1997) proposes a gradient ascent, based upon a Monte Carlo procedure for estimating $E_\theta(f_j)$. The idea is to generate random samples of feature structures from the distribution $P_{\hat{\theta}}(\omega)$, where $\hat{\theta}$ is the current parameter estimate, and to use these to estimate $E_{\hat{\theta}}(f_j)$, and hence the gradient of the likelihood. Samples are generated as follows: Given a SUBG, Abney constructs a covering PCFG based upon the SUBG and $\hat{\theta}$, the current estimate of θ . The derivation trees of the PCFG can be mapped onto a set containing all of the SUBG's syntactic analyses. Monte Carlo samples from the PCFG are comparatively easy to generate, and sample syntactic analyses that do not map to well-formed SUBG syntactic structures are then simply discarded. This generates a stream of syntactic structures, but not distributed according to $P_{\hat{\theta}}(\omega)$ (distributed instead according to the restriction of the PCFG to the SUBG). Abney proposes using a Metropolis acceptance-rejection method to adjust the distribution of this stream of feature structures to achieve detailed balance, which then produces a stream of feature structures distributed according to $P_{\hat{\theta}}(\omega)$.

While this scheme is theoretically sound, it would appear to be computationally impracti-

cal for realistic SUBGs. Every step of the proposed procedure (corresponding to a single step of gradient ascent) requires a very large number of PCFG samples: samples must be found that correspond to well-formed SUBGs; many such samples are required to bring the Metropolis algorithm to (near) equilibrium; many samples are needed at equilibrium to properly estimate $E_{\hat{\theta}}(f_j)$.

The idea of a gradient ascent of the likelihood (2) is appealing—a simple calculation reveals that the likelihood is concave and therefore free of local maxima. But the gradient (in particular, $E_\theta(f_j)$) is intractable. This motivates an alternative strategy involving a data-based estimate of $E_\theta(f_j)$:

$$\begin{aligned} E_\theta(f_j) &= E_\theta(E_\theta(f_j(\omega)|y(\omega))) \quad (4) \\ &\approx \frac{1}{n} \sum_{i=1, \dots, n} E_\theta(f_j(\omega)|y(\omega) = y_i) \quad (5) \end{aligned}$$

where $y(\omega)$ is the yield belonging to the syntactic analysis ω , and $y_i = y(\omega_i)$ is the yield belonging to the i 'th sample in the training corpus.

The point is that $E_\theta(f_j(\omega)|y(\omega) = y_i)$ is generally computable. In fact, if $\Omega(y)$ is the set of well-formed syntactic structures that have yield y (i.e., the set of possible parses of the string y), then

$$\begin{aligned} E_\theta(f_j(\omega)|y(\omega) = y_i) &= \\ &= \frac{\sum_{\omega' \in \Omega(y_i)} f_j(\omega') e^{\sum_{k=1, \dots, m} \theta_k f_k(\omega')}}{\sum_{\omega' \in \Omega(y_i)} e^{\sum_{k=1, \dots, m} \theta_k f_k(\omega')}} \end{aligned}$$

Hence the calculation of the conditional expectations only involves summing over the possible syntactic analyses or parses $\Omega(y_i)$ of the strings in the training corpus. While it is possible to construct UBGs for which the number of possible parses is unmanageably high, for many grammars it is quite manageable to enumerate the set of possible parses and thereby directly evaluate $E_\theta(f_j(\omega)|y(\omega) = y_i)$.

Therefore, we propose replacing the gradient, (3), by

$$f_j(\tilde{\omega}) - \sum_{i=1, \dots, n} E_\theta(f_j(\omega)|y(\omega) = y_i) \quad (6)$$

and performing a gradient ascent. Of course (6) is no longer the gradient of the likelihood func-

tion, but fortunately it is (exactly) the gradient of (the log of) another criterion:

$$\text{PL}_\theta(\tilde{\omega}) = \prod_{i=1, \dots, n} P_\theta(\omega = \omega_i | y(\omega) = y_i) \quad (7)$$

Instead of maximizing the likelihood of the syntactic analyses over the training corpus, we maximize the *conditional* likelihood of these analyses *given the observed yields*. In our experiments, we have used a conjugate-gradient optimization program adapted from the one presented in Press et al. (1992).

Regardless of the pragmatic (computational) motivation, one could perhaps argue that the conditional probabilities $P_\theta(\omega|y)$ are as useful (if not more useful) as the full probabilities $P_\theta(\omega)$, at least in those cases for which the ultimate goal is syntactic analysis. Berger et al. (1996) and Jelinek (1997) make this same point and arrive at the same estimator, albeit through a maximum entropy argument.

The problem of estimating parameters for log-linear models is not new. It is especially difficult in cases, such as ours, where a large sample space makes the direct computation of expectations infeasible. Many applications in spatial statistics, involving Markov random fields (MRF), are of this nature as well. In his seminal development of the MRF approach to spatial statistics, Besag introduced a “pseudo-likelihood” estimator to address these difficulties (Besag, 1974; Besag, 1975), and in fact our proposal here is an instance of his method. In general, the likelihood function is replaced by a more manageable product of conditional likelihoods (a *pseudo-likelihood*—hence the designation PL_θ), which is then optimized over the parameter vector, instead of the likelihood itself. In many cases, as in our case here, this substitution side-steps much of the computational burden *without sacrificing consistency* (more on this shortly).

What are the asymptotics of optimizing a pseudo-likelihood function? Look first at the likelihood itself. For large n :

$$\begin{aligned} \frac{1}{n} \log L_\theta(\tilde{\omega}) &= \frac{1}{n} \log \prod_{i=1, \dots, n} P_\theta(\omega_i) \\ &= \frac{1}{n} \sum_{i=1, \dots, n} \log P_\theta(\omega_i) \end{aligned}$$

$$\approx \int P_{\theta_o}(\omega) \log P_\theta(\omega) d\omega \quad (8)$$

where θ_o is the true (and unknown) parameter vector. Up to a constant, (8) is the negative of the Kullback-Leibler divergence between the true and estimated distributions of syntactic analyses. As sample size grows, maximizing likelihood amounts to minimizing divergence. As for pseudo-likelihood:

$$\begin{aligned} \frac{1}{n} \log \text{PL}_\theta(\tilde{\omega}) &= \frac{1}{n} \log \prod_{i=1, \dots, n} P_\theta(\omega = \omega_i | y(\omega) = y_i) \\ &= \frac{1}{n} \sum_{i=1, \dots, n} \log P_\theta(\omega = \omega_i | y(\omega) = y_i) \\ &\approx E_{\theta_o} \left[\int P_{\theta_o}(\omega|y) \log P_\theta(\omega|y) d\omega \right] \end{aligned}$$

So that maximizing pseudo-likelihood (at large samples) amounts to minimizing the average (over yields) divergence between the true and estimated *conditional distributions of analyses given yields*.

Maximum likelihood estimation is consistent: under broad conditions the sequence of distributions $P_{\hat{\theta}_n}$, associated with the maximum likelihood estimator for θ_o given the samples $\omega_1, \dots, \omega_n$, converges to P_{θ_o} . Pseudo-likelihood is also consistent, but in the present implementation it is consistent for the conditional distributions $P_{\theta_o}(\omega|y(\omega))$ and not necessarily for the full distribution P_{θ_o} (see Chi (1998)). It is not hard to see that pseudo-likelihood will not always correctly estimate P_{θ_o} . Suppose there is a feature f_i which depends only on yields: $f_i(\omega) = f_i(y(\omega))$. (Later we will refer to such features as *pseudo-constant*.) In this case, the derivative of $\text{PL}_\theta(\tilde{\omega})$ with respect to θ_i is zero; $\text{PL}_\theta(\tilde{\omega})$ contains no information about θ_i . In fact, in this case *any* value of θ_i gives the *same* conditional distribution $P_\theta(\omega|y(\omega))$; θ_i is irrelevant to the problem of choosing good parses.

Despite the assurance of consistency, pseudo-likelihood estimation is prone to over fitting when a large number of features is matched against a modest-sized training corpus. One particularly troublesome manifestation of over fitting results from the existence of features which, relative to the training set, we might term “pseudo-maximal”: Let us say that a feature f is *pseudo-maximal* for a yield y iff

$\forall \omega' \in \Omega(y) f(\omega) \geq f(\omega')$ where ω is any correct parse of y , i.e., the feature's value on every correct parse ω of y is greater than or equal to its value on any other parse of y . Pseudo-minimal features are defined similarly. It is easy to see that if f_j is pseudo-maximal on *each sentence* of the training corpus then the parameter assignment $\theta_j = \infty$ maximizes the corpus pseudo-likelihood. (Similarly, the assignment $\theta_j = -\infty$ maximizes pseudo-likelihood if f_j is pseudo-minimal over the training corpus). Such infinite parameter values indicate that the model treats pseudo-maximal features categorically; i.e., any parse with a non-maximal feature value is assigned a zero conditional probability.

Of course, a feature which is pseudo-maximal over the training corpus is not necessarily pseudo-maximal for all yields. This is an instance of over fitting, and it can be addressed, as is customary, by adding a regularization term that promotes small values of θ to the objective function. A common choice is to add a quadratic to the log-likelihood, which corresponds to multiplying the likelihood itself by a normal distribution. In our experiments, we multiplied the pseudo-likelihood by a zero-mean normal in $\theta_1, \dots, \theta_m$, with diagonal covariance, and with standard deviation σ_j for θ_j equal to 7 times the maximum value of f_j found in any parse in the training corpus. (We experimented with other values for σ_j , but the choice seems to have little effect). Thus instead of maximizing the log pseudo-likelihood, we choose $\hat{\theta}$ to maximize

$$\log \text{PL}_{\theta}(\tilde{\omega}) - \sum_{j=1, \dots, m} \frac{\theta_j^2}{2\sigma_j^2} \quad (9)$$

4 A maximum correct estimator for log linear models

The pseudo-likelihood estimator described in the last section finds parameter values which maximize the conditional probabilities of the observed parses (syntactic analyses) given the observed sentences (yields) in the training corpus. One of the empirical evaluation measures we use in the next section measures the number of correct parses selected from the set of all possible parses. This suggests another possible objective function: choose $\hat{\theta}$ to maximize the number $C_{\hat{\theta}}(\tilde{\omega})$ of times the maximum likelihood parse (under θ) is in fact the correct parse,

in the training corpus.

$C_{\theta}(\tilde{\omega})$ is a highly discontinuous function of θ , and most conventional optimization algorithms perform poorly on it. We had the most success with a slightly modified version of the simulated annealing optimizer described in Press et al. (1992). This procedure is much more computationally intensive than the gradient-based pseudo-likelihood procedure. Its computational difficulty grows (and the quality of solutions degrade) rapidly with the number of features.

5 Empirical evaluation

Ron Kaplan and Hadar Shemtov at Xerox PARC provided us with two LFG parsed corpora. The Verbmobil corpus contains appointment planning dialogs, while the Homecentre corpus is drawn from Xerox printer documentation. Table 1 summarizes the basic properties of these corpora. These corpora contain packed c/f-structure representations (Maxwell III and Kaplan, 1995) of the grammatical parses of each sentence with respect to Lexical-Functional grammars. The corpora also indicate which of these parses is in fact the correct parse (this information was manually entered). Because slightly different grammars were used for each corpus we chose not to combine the two corpora, although we used the set of features described in section 2 for both in the experiments described below. Table 2 describes the properties of the features used for each corpus.

In addition to the two estimators described above we also present results from a baseline estimator in which all parses are treated as equally likely (this corresponds to setting all the parameters θ_j to zero).

We evaluated our estimators using held-out test corpus $\tilde{\omega}_{\text{test}}$. We used two evaluation measures. In an actual parsing application a SUBG might be used to identify the correct parse from the set of grammatical parses, so our first evaluation measure counts the number $C_{\hat{\theta}}(\tilde{\omega}_{\text{test}})$ of sentences in the test corpus $\tilde{\omega}_{\text{test}}$ whose maximum likelihood parse under the estimated model $\hat{\theta}$ is actually the correct parse. If a sentence has l most likely parses (i.e., all l parses have the same conditional probability) and one of these parses is the correct parse, then we score $1/l$ for this sentence.

The second evaluation measure is the pseudo-

	Verbmobil corpus	Homecentre corpus
Number of sentences	540	980
Number of ambiguous sentences	314	481
Number of parses of ambiguous sentences	3245	3169

Table 1: Properties of the two corpora used to evaluate the estimators.

	Verbmobil corpus	Homecentre corpus
Number of features	191	227
Number of rule features	59	57
Number of pseudo-constant features	19	41
Number of pseudo-maximal features	12	4
Number of pseudo-minimal features	8	5

Table 2: Properties of the features used in the stochastic LFG models. The numbers of pseudo-maximal and pseudo-minimal features do not include pseudo-constant features.

likelihood itself, $PL_{\hat{\theta}}(\tilde{\omega}_{\text{test}})$. The pseudo-likelihood of the test corpus is the likelihood of the correct parses given their yields, so pseudo-likelihood measures how much of the probability mass the model puts onto the correct analyses. This metric seems more relevant to applications where the system needs to estimate how likely it is that the correct analysis lies in a certain set of possible parses; e.g., ambiguity-preserving translation and human-assisted disambiguation. To make the numbers more manageable, we actually present the negative logarithm of the pseudo-likelihood rather than the pseudo-likelihood itself—so smaller is better.

Because of the small size of our corpora we evaluated our estimators using a 10-way cross-validation paradigm. We randomly assigned sentences of each corpus into 10 approximately equal-sized subcorpora, each of which was used in turn as the test corpus. We evaluated on each subcorpus the parameters that were estimated from the 9 remaining subcorpora that served as the training corpus for this run. The evaluation scores from each subcorpus were summed in order to provide the scores presented here.

Table 3 presents the results of the empirical evaluation. The superior performance of both estimators on the Verbmobil corpus probably reflects the fact that the non-rule features were designed to match both the grammar and content of that corpus. The pseudo-likelihood estimator performed better than the correct-parses estimator on both corpora un-

der both evaluation metrics. There seems to be substantial over learning in all these models; we routinely improved performance by discarding features. With a small number of features the correct-parses estimator typically scores better than the pseudo-likelihood estimator on the correct-parses evaluation metric, but the pseudo-likelihood estimator always scores better on the pseudo-likelihood evaluation metric.

6 Conclusion

This paper described a log-linear model for SUBGs and evaluated two estimators for such models. Because estimators that can estimate rule features for SUBGs can also estimate other kinds of features, there is no particular reason to limit attention to rule features in a SUBG. Indeed, the number and choice of features strongly influences the performance of the model. The estimated models are able to identify the correct parse from the set of all possible parses approximately 50% of the time.

We would have liked to introduce features corresponding to dependencies between lexical items. Log-linear models are well-suited for lexical dependencies, but because of the large number of such dependencies substantially larger corpora will probably be needed to estimate such models.¹

¹Alternatively, it may be possible to use a simpler non-SUBG model of lexical dependencies estimated from a much larger corpus as the reference distribution with

	Verbmobil corpus		Homecentre corpus	
	$C(\tilde{\omega}_{\text{test}})$	$-\log \text{PL}(\tilde{\omega}_{\text{test}})$	$C(\tilde{\omega}_{\text{test}})$	$-\log \text{PL}(\tilde{\omega}_{\text{test}})$
Baseline estimator	9.7%	533	15.2%	655
Pseudo-likelihood estimator	58.7%	396	58.8%	583
Correct-parses estimator	53.7%	469	53.2%	604

Table 3: An empirical evaluation of the estimators. $C(\tilde{\omega}_{\text{test}})$ is the number of maximum likelihood parses of the test corpus that were the correct parses, and $-\log \text{PL}(\tilde{\omega}_{\text{test}})$ is the negative logarithm of the pseudo-likelihood of the test corpus.

However, there may be applications which can benefit from a model that performs even at this level. For example, in a machine-assisted translation system a model like ours could be used to order possible translations so that more likely alternatives are presented before less likely ones. In the ambiguity-preserving translation framework, a model like this one could be used to choose between sets of analyses whose ambiguities cannot be preserved in translation.

References

- Steven P. Abney. 1997. Stochastic Attribute-Value Grammars. *Computational Linguistics*, 23(4):597–617.
- Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.
- J. Besag. 1974. Spatial interaction and the statistical analysis of lattice systems (with discussion). *Journal of the Royal Statistical Society, Series D*, 36:192–236.
- J. Besag. 1975. Statistical analysis of non-lattice data. *The Statistician*, 24:179–195.
- Zhiyi Chi. 1998. *Probability Models for Complex Systems*. Ph.D. thesis, Brown University.
- Brendan J. Frey. 1998. *Graphical Models for Machine Learning and Digital Communication*. The MIT Press, Cambridge, Massachusetts.
- Jerry R. Hobbs and John Bear. 1995. Two principles of parse preference. In Antonio Zampolli, Nicoletta Calzolari, and Martha Palmer, editors, *Linguistica Computazionale: Current Issues in Computational Linguistics: In Honour of Don Walker*, pages 503–512. Kluwer.
- Frederick Jelinek. 1997. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, Massachusetts.
- John T. Maxwell III and Ronald M. Kaplan. 1995. A method for disjunctive constraint satisfaction. In Mary Dalrymple, Ronald M. Kaplan, John T. Maxwell III, and Annie Zaenen, editors, *Formal Issues in Lexical-Functional Grammar*, number 47 in CSLI Lecture Notes Series, chapter 14, pages 381–481. CSLI Publications.
- Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California.
- William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 1992. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, 2nd edition.

respect to which the SUBG model is defined, as described in Jelinek (1997).

Estimation of Stochastic Attribute-Value Grammars using an Informative Sample

Miles Osborne

osborne@let.rug.nl

Rijksuniversiteit Groningen, The Netherlands*

Abstract

We argue that some of the computational complexity associated with estimation of stochastic attribute-value grammars can be reduced by training upon an informative subset of the full training set. Results using the parsed Wall Street Journal corpus show that in some circumstances, it is possible to obtain better estimation results using an informative sample than when training upon all the available material. Further experimentation demonstrates that with unlexicalised models, a Gaussian prior can reduce overfitting. However, when models are lexicalised and contain overlapping features, overfitting does not seem to be a problem, and a Gaussian prior makes minimal difference to performance. Our approach is applicable for situations when there are an infeasibly large number of parses in the training set, or else for when recovery of these parses from a packed representation is itself computationally expensive.

1 Introduction

Abney showed that attribute-value grammars cannot be modelled adequately using statistical techniques which assume that statistical dependencies are accidental (Abney, 1997). Instead of using a model class that assumed independence, Abney suggested using *Random Fields Models* (RFMs) for attribute-value grammars. RFMs deal with the graphical structure of a parse. Because they do not make independence assumptions about the stochastic generation process that might have produced some parse, they are able to model correctly dependencies that exist within parses.

When estimating standardly-formulated RFMs, it is necessary to sum over all parses licensed by the grammar. For many broad coverage natural language grammars, this might involve summing over an exponential number of parses. This would make the task computationally intractable. Abney, following the lead of Lafferty *et al*, suggested a Monte

Carlo simulation as a way of reducing the computational burden associated with RFM estimation (Lafferty *et al*, 1997). However, Johnson *et al* considered the form of sampling used in this simulation (Metropolis-Hastings) intractable (Johnson *et al*, 1999). Instead, they proposed an alternative strategy that redefined the estimation task. It was argued that this redefinition made estimation computationally simple enough that a Monte Carlo simulation was unnecessary. They presented results obtained using a small unlexicalised model trained on a modest corpus.

Unfortunately, Johnson *et al* assumed it was possible to retrieve *all* parses licensed by a grammar when parsing a given training set. For us, this was not the case. In our experiments with a manually written broad coverage Definite Clause Grammar (DCG) (Briscoe and Carroll, 1996), we were only able to recover all parses for Wall Street Journal sentences that were at most 13 tokens long within acceptable time and space bounds on computation. When we used an incremental Minimum Description Length (MDL) based learner to extend the coverage of our manually written grammar (from roughly 60% to around 90% of the parsed Wall Street Journal), the situation became worse. Sentence ambiguity considerably increased. We were then only able to recover all parses for Wall Street Journal sentences that were at most 6 tokens long (Osborne, 1999).

We *can* however, and usually in polynomial time, recover up to 30 parses for sentences up to 30 tokens long when we use a probabilistic unpacking mechanism (Carroll and Briscoe, 1992). (Longer sentences than 30 tokens can be parsed, but the number of parses we can recover for them drops off rapidly).¹ However, 30 is far less than the maximum number

¹We made an attempt to determine the maximum number of parses our grammar might assign to sentences. On a 450MHz Ultra Sparc 80 with 2 Gb of real memory, with a limit of at most 1000 parses per sentence, and allowing no more than 100 CPU seconds per sentence, we found that sentence ambiguity increased exponentially with respect to sentence length. Sentences with 30 tokens had an estimated average of 866 parses (standard deviation 290.4). Without the limit of 1000 parses per sentence, it seems likely that this average would increase.

* Current address: osborne@cogsci.ed.ac.uk, University of Edinburgh, Division of Informatics, 2 Buccleuch Place, EH8 9LW, Scotland.

of parses per sentence our grammar might assign to Wall Street Journal sentences. Any training set we have access to will therefore be necessarily limited in size.

We therefore need an estimation strategy that takes seriously the issue of extracting the best performance from a limited size training set. A limited size training set means one created by retrieving at most n parses per sentence. Although we cannot recover all possible parses, we do have a choice as to which parses estimation should be based upon.

Our approach to the problem of making RFM estimation feasible for our highly ambiguous DCG is to seek out an *informative* sample and train upon that. We do not redefine the estimation task in a non-standard way, nor do we use a Monte Carlo simulation.

We call a sample *informative* if it both leads to the selection of a model that does not underfit or overfit, and also is typical of future samples. Despite one’s intuitions, an informative sample might be a proper subset of the full training set. This means that estimation using the informative sample might yield better results than estimation using all of the training set.

The rest of this paper is as follows. Firstly we introduce RFMs. Then we show how they may be estimated and how an informative sample might be identified. Next, we give details of the attribute-value grammar we use, and show how we go about modelling it. We then present two sets of experiments. The first set is small scale, and are designed to show the existence of an informative sample. The second set of experiments are larger in scale, and build upon the computational savings we are able to achieve using a probabilistic unpacking strategy. They show how large models (two orders of magnitude larger than those reported by Johnson *et al*) can be estimated using the parsed Wall Street Journal corpus. Overfitting is shown to take place. They also show how this overfitting can be (partially) reduced by using a Gaussian prior. Finally, we end with some comments on our work.

2 Random Field Models

Here we show how attribute-value grammars may be modelled using RFMs. Although our commentary is in terms of RFMs and grammars, it should be obvious that RFM technology can be applied to other estimation scenarios.

Let G be an attribute-value grammar, D the set of sentences within the string-set defined by $L(G)$ and Ω the union of the set of parses assigned to each sentence in D by the grammar G . A Random Field Model, M , consist of two components: a set of *features*, F and a set of *weights*, Λ .

Features are the basic building blocks of RFMs.

They enable the system designer to specify the key aspects of what it takes to differentiate one parse from another parse. Each feature is a function from a parse to an integer. Here, the integer value associated with a feature is interpreted as the number of times a feature ‘matches’ (is ‘active’) with a parse. Note features should not be confused with features as found in feature-value bundles (these will be called attributes instead). Features are usually manually selected by the system designer.

The other component of a RFM, Λ , is a set of weights. Informally, weights tell us how features are to be used when modelling parses. For example, an active feature with a large weight might indicate that some parse had a high probability. Each weight λ_i is associated with a feature f_i . Weights are real-valued numbers and are automatically determined by an estimation process (for example using Improved Iterative Scaling (Lafferty et al., 1997)). One of the nice properties of RFMs is that the likelihood function of a RFM is strictly concave. This means that there are no local minima, and so we can be sure that scaling will result in estimation of a RFM that is globally optimal.

The (unnormalised) total weight of a parse x , $\psi(x)$, is a function of the k features that are ‘active’ on a parse:

$$\psi(x) = \exp\left(\sum_{i=1}^k \lambda_i f_i(x)\right) \quad (1)$$

The probability of a parse, $P(x | M)$, is simply the result of normalising the total weight associated with that parse:

$$P(x | M) = \frac{1}{Z} \psi(x) \quad (2)$$

$$Z = \sum_{y \in \Omega} \psi(y) \quad (3)$$

The interpretation of this probability depends upon the application of the RFM. Here, we use parse probabilities to reflect preferences for parses.

When using RFMs for parse selection, we simply select the parse that maximises $\psi(x)$. In these circumstances, there is no need to normalise (compute Z). Also, when computing $\psi(x)$ for competing parses, there is no built-in bias towards shorter (or longer) derivations, and so no need to normalise with respect to derivation length.²

²The reason there is no need to normalise with respect to derivation length is that features can have positive or negative weights. The weight of a parse will therefore not always monotonically increase with respect to the number of active features.

3 RFM Estimation and Selection of the Informative Sample

We now sketch how RFMs may be estimated and then outline how we seek out an informative sample.

We use Improved Iterative Scaling (IIS) to estimate RFMs. In outline, the IIS algorithm is as follows:

1. Start with a reference distribution R , a set of features F and a set of weights Λ . Let M be the RFM defined using F and Λ .
2. Initialise all weights to zero. This makes the initial model uniform.
3. Compute the expectation of each feature w.r.t R .
4. For each feature f_i
 - (a) Find a weight $\check{\lambda}_i$ that equates the expectation of f_i w.r.t R and the expectation of f_i w.r.t M .
 - (b) Replace the old value of λ_i with $\check{\lambda}_i$.
5. If the model has converged to R , output M .
6. Otherwise, go to step 4

The key step here is 4a, computing the expectations of features w.r.t the RFM. This involves calculating the probability of a parse, which, as we saw from equation 2, requires a summation over all parses in Ω .

We seek out an informative sample Ω_t ($\Omega_t \subseteq \Omega$) as follows:

1. Pick out from Ω a sample of size n .
2. Estimate a model using that sample and evaluate it.
3. If the model just estimated shows signs of overfitting (with respect to an unseen held-out data set), halt and output the model.
4. Otherwise, increase n and go back to step 1.

Our approach is motivated by the following (partially related) observations:

- Because we use a non-parametric model class and select an instance of it in terms of some sample (section 5 gives details), a stochastic complexity argument tells us that an overly simple model (resulting from a small sample) is likely to underfit. Likewise, an overly complex model (resulting from a large sample) is likely to overfit. An informative sample will therefore relate to a model that does not under or overfit.
- On average, an informative sample will be ‘typical’ of future samples. For many real-life situations, this set is likely to be small relative to the size of the full training set.

We incorporate the first observation through our search mechanism. Because we start with small samples and gradually increase their size, we remain within the domain of efficiently recoverable samples.

The second observation is (largely) incorporated in the way we pick samples. The experimental section of this paper goes into the relevant details.

Note our approach is heuristic: we cannot afford to evaluate all $2^{|\Omega|}$ possible training sets. The actual size of the informative sample Ω_t will depend both upon the model class used and the maximum sentence length we can deal with. We would expect richer, lexicalised models to exhibit overfitting with smaller samples than would be the case with unlexicalised models. We would expect the size of an informative sample to increase as the maximum sentence length increased.

There are similarities between our approach and with estimation using MDL (Rissanen, 1989). However, our implementation does not explicitly attempt to minimise code lengths. Also, there are similarities with importance sampling approaches to RFM estimation (such as (Chen and Rosenfeld, 1999a)). However, such attempts do not minimise under or overfitting.

4 The Grammar

The grammar we model with Random Fields, (called the *Tag Sequence Grammar* (Briscoe and Carroll, 1996), or TSG for short) was developed with regard to coverage, and when compiled consists of 455 Definite Clause Grammar (DCG) rules. It does not parse sequences of words directly, but instead assigns derivations to sequences of part-of-speech tags (using the CLAWS2 tagset. The grammar is relatively shallow, (for example, it does not fully analyse unbounded dependencies) but it does make an attempt to deal with common constructions, such as dates or names, commonly found in corpora, but of little theoretical interest. Furthermore, it integrates into the syntax a text grammar, grouping utterances into units that reduce the overall ambiguity.

5 Modelling the Grammar

Modelling the TSG with respect to the parsed Wall Street Journal consists of two steps: creation of a feature set and definition of the reference distribution.

Our feature set is created by parsing sentences in the training set (Ω_T), and using each parse to instantiate *templates*. Each template defines a family of features. At present, the templates we use are somewhat ad-hoc. However, they are motivated by the observations that linguistically-stipulated units (DCG rules) are informative, and that many DCG applications in preferred parses can be predicted using lexical information.

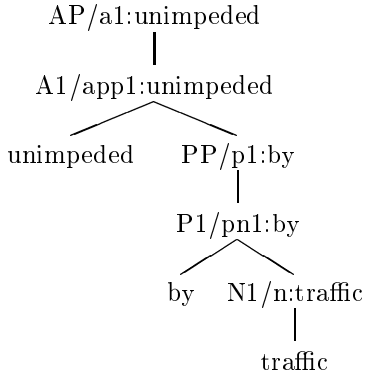


Figure 1: TSG Parse Fragment

The first template creates features that count the number of times a DCG instantiation is present within a parse.³ For example, suppose we parsed the Wall Street Journal AP:

1 unimpeded by traffic

A parse tree generated by TSG might be as shown in figure 1. Here, to save on space, we have labelled each interior node in the parse tree with TSG rule names, and not attribute-value bundles. Furthermore, we have annotated each node with the head word of the phrase in question. Within our grammar, heads are (usually) explicitly marked. This means we do not have to make any guesses when identifying the head of a local tree. With head information, we are able to lexicalise models. We have suppressed tagging information.

For example, a feature defined using this template might count the number of times the we saw:

$$\begin{array}{c} \text{AP/a1} \\ | \\ \text{A1/app1} \end{array}$$

in a parse. Such features record some of the context of the rule application, in that rule applications that differ in terms of how attributes are bound will be modelled by different features.

Our second template creates features that are partially lexicalised. For each local tree (of depth one) that has a PP daughter, we create a feature that counts the number of times that local tree, decorated with the head-word of the PP, was seen in a parse. An example of such a lexicalised feature would be:

$$\begin{array}{c} \text{A1/app1} \\ | \\ \text{PP/p1:by} \end{array}$$

³Note, all our features suppress any terminals that appear in a local tree. Lexical information is included when we decide to lexicalise features.

These features are designed to model PP attachments that can be resolved using the head of the PP.

The third and final template creates features that are again partially lexicalised. This time, we create local trees of depth one that are decorated with the head word. For example, here is one such feature:

$$\begin{array}{c} \text{AP/a1:unimpeded} \\ | \\ \text{A1/app1} \end{array}$$

Note the second and third templates result in features that overlap with features resulting from applications of the first template.

We create the reference distribution R (an association of probabilities with TSG parses of sentences, such that the probabilities reflect parse preferences) using the following process:

1. Extract some sample Ω_T (using the approach mentioned in section 3).
2. For each sentence in the sample, for each parse of that sentence, compute the ‘distance’ between the TSG parse and the WSJ reference parse. In our approach, distance is calculated in terms of a weighted sum of crossing rates, recall and precision. Minimising it maximises our definition of parse plausibility.⁴ However, there is nothing inherently crucial about this decision. Any other objective function (that can be represented as an exponential distribution) could be used instead.
3. Normalise the distances, such that for some sentence, the sum of the distances of all recovered TSG parses for that sentence is a constant across all sentences. Normalising in this manner ensures that each sentence is equiprobable (remember that RFM probabilities are in terms of parse preferences, and not probability of occurrence in some corpus).
4. Map the normalised distances into probabilities. If $d(p)$ is the normalised distance of TSG parse p , then associate with parse p the reference probability given by the maximum likelihood estimator:

$$\frac{d(p)}{\sum_{x \in \Omega_t} d(x)} \quad (4)$$

Our approach therefore gives partial credit (a non-zero reference probability) to all parses in Ω_t . R is therefore not as discontinuous as the equivalent distribution used by Johnson *et al.* We therefore do not need to use simulated annealing or other numerically intensive techniques to estimate models.

⁴Our distance metric is the same one used by Hektoen (Hektoen, 1997)

6 Experiments

Here we present two sets of experiments. The first set demonstrate the existence of an informative sample. It also shows some of the characteristics of three sampling strategies. The second set of experiments is larger in scale, and show RFMs (both lexicalised and unlexicalised) estimated using sentences up to 30 tokens long. Also, the effects of a Gaussian prior are demonstrated as a way of (partially) dealing with overfitting.

6.1 Testing the Various Sampling Strategies

In order to see how various sizes of sample related to estimation accuracy and whether we could achieve similar levels of performance without recovering all possible parses, we ran the following experiments.

We used a model consisting of features that were defined using all three templates. We also threw away all features that occurred less than two times in the training set. We randomly split the Wall Street Journal into disjoint training, held-out and testing sets. All sentences in the training and held-out sets were at most 14 tokens long. Sentences in the testing set were at most 30 tokens long. There were 6626 sentences in the training set, 98 sentences in the held-out set and 441 sentences in the testing set. Sentences in the held-out set had on average 12.6 parses, whilst sentences in the testing-set had on average 60.6 parses per sentence.

The held-out set was used to decide which model performed best. Actual performance of the models should be judged with respect to the testing set.

Evaluation was in terms of exact match: for each sentence in the test set, we awarded ourselves a point if the RFM ranked highest the same parse that was ranked highest using the reference probabilities. When evaluating with respect to the held-out set, we recovered all parses for sentences in the held-out set. When evaluating with respect to the testing-set, we recovered at most 100 parses per sentence.

For each run, we ran IIS for the same number of iterations (20). In each case, we evaluated the RFM after each other iteration and recorded the best classification performance. This step was designed to avoid overfitting distorting our results.

Figure 2 shows the results we obtained with possible ways of picking ‘typical’ samples. The first column shows the maximum number of parses per sentences that we retrieved in each sample.

The second column shows the size of the sample (in parses).

The other columns give classification accuracy results (a percentage) with respect to the testing set. In parentheses, we give performance with respect to the held-out set.

The column marked *Rand* shows the performance

Max parses	Size	Rand	SCFG	Ref
1	6626	25.2 (51.7)	23.3 (50.0)	23.4 (50.0)
2	12331	37.9 (63.0)	40.4 (60.3)	40.4 (60.0)
3	17026	43.2 (65.5)	43.7 (63.8)	43.7 (63.8)
5	24878	43.7 (70.2)	45.8 (69.5)	45.8 (69.5)
10	39581	47.4 (72.0)	47.0 (70.0)	46.9 (70.0)
100	119694	45.0 (68.7)	45.0 (68.0)	45.0 (68.0)
1000	246686	44.4 (67.4)	43.0 (67.0)	43.0 (67.0)
∞	267400	43.0 (66.0)	43.0 (66.0)	43.0 (66.0)

Figure 2: Results with various sampling strategies

of runs that used a sample that contained parses which were randomly and uniformly selected out of the set of all possible parses. The classification accuracy results for this sampler are averaged over 10 runs.

The column marked *SCFG* shows the results obtained when using a sample that contained parses that were retrieved using the probabilistic unpacking strategy. This did not involve retrieving all possible parses for each sentence in the training set. Since there is no random component, the results are from a single run. Here, parses were ranked using a stochastic context free backbone approximation of TSG. Parameters were estimated using simple counting.

Finally, the column marked *Ref* shows the results obtained when using a sample that contained the overall n -best parses per sentence, as defined in terms of the reference distribution.

As a baseline, a model containing randomly assigned weights produced a classification accuracy of 45% on the held-out sentences. These results were averaged over 10 runs.

As can be seen, increasing the sample size produces better results (for each sampling strategy). Around a sample size of 40k parses, overfitting starts to manifest, and performance bottoms-out. One of these is therefore our informative sample. Note that the best sample (40k parses) is less than 20% of the total possible training set.

The difference between the various samplers is marginal, with a slight preference for *Rand*. However the fact that *SCFG* sampling seems to do almost as well as *Rand* sampling, and furthermore does not require unpacking all parses, makes it the sampling strategy of choice.

SCFG sampling is biased in the sense that the sample produced using it will tend to concentrate around those parses that are all close to the best parses. *Rand* sampling is unbiased, and, apart from the practical problems of having to recover all parses, might in some circumstances be better than *SCFG* sampling. At the time of writing this paper, it was unclear whether we could combine *SCFG* with *Rand* sampling -sample parses from the full distribu-

tion without unpacking all parses. We suspect that for probabilistic unpacking to be efficient, it must rely upon some non-uniform distribution. Unpacking randomly and uniformly would probably result in a large loss in computational efficiency.

6.2 Larger Scale Evaluation

Here we show results using a larger sample and testing set. We also show the effects of lexicalisation, overfitting, and overfitting avoidance using a Gaussian prior. Strictly speaking this section could have been omitted from the paper. However, if one views estimation using an informative sample as overfitting avoidance, then estimation using a Gaussian prior can be seen as another, complementary take on the problem.

The experimental setup was as follows. We randomly split the Wall Street Journal corpus into a training set and a testing set. Both sets contained sentences that were at most 30 tokens long. When creating the set of parses used to estimate RFMs, we used the SCFG approach, and retained the top 25 parses per sentence. Within the training set (arising from 16,200 sentences), there were 405,020 parses. The testing set consisted of 466 sentences, with an average of 60.6 parses per sentence.

When evaluating, we retrieved at most 100 parses per sentence in the testing set and scored them using our reference distribution. As before, we awarded ourselves a point if the most probable testing parse (in terms of the RMF) coincided with the most probable parse (in terms of the reference distribution). In all cases, we ran IIS for 100 iterations.

For the first experiment, we used just the first template (features that related to DCG instantiations) to create model 1; the second experiment used the first and second templates (additional features relating to PP attachment) to create model 2. The final experiment used all three templates (additional features that were head-lexicalised) to create model 3.

The three models contained 39,230, 65,568 and 278,127 features respectively,

As a baseline, a model containing randomly assigned weights achieved a 22% classification accuracy. These results were averaged over 10 runs. Figure 3 shows the classification accuracy using models 1, 2 and 3.

As can be seen, the larger scale experimental results were better than those achieved using the smaller samples (mentioned in section 6.1). The reason for this was because we used longer sentences. The informative sample derivable from such a training set was likely to be larger (more representative of

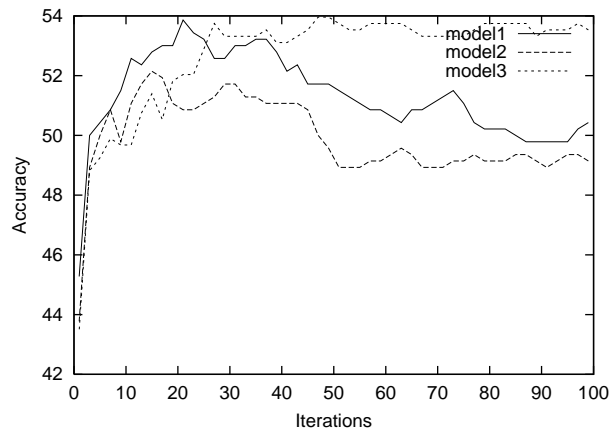


Figure 3: Classification Accuracy for Three Models Estimated using Basic IIS

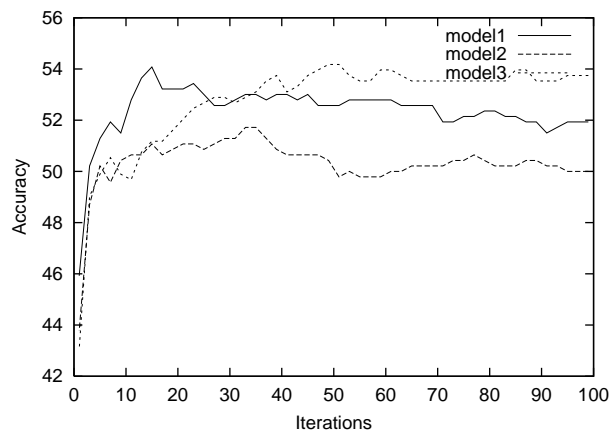


Figure 4: Classification Accuracy for Three Models Estimated using a Gaussian Prior and IIS

the population) than the informative sample derivable from a training set using shorter, less syntactically complex sentences. With the unlexicalised model, we see clear signs of overfitting. Model 2 overfits even more so. For reasons that are unclear, we see that the larger model 3 does not appear to exhibit overfitting.

We next used the Gaussian Prior method of Chen and Rosenfeld to reduce overfitting (Chen and Rosenfeld, 1999b). This involved integrating a Gaussian prior (with a zero mean) into IIS and searching for the model that maximised the product of the likelihood and prior probabilities. For the experiments reported here, we used a single variance over the entire model (better results might be achievable if multiple variances were used, perhaps with one variance per template type). The actual value of the variance was found by trial-and-error. However, optimisation using a held-out set is easy to achieve.

We repeated the large-scale experiment, but this time using a Gaussian prior. Figure 4 shows the classification accuracy of the models when using a Gaussian Prior.

When we used a Gaussian prior, we found that all models showed signs of improvement (albeit with varying degrees): performance either increased, or else did not decrease with respect to the number of iterations. Still, model 2 continued to underperform. Model 3 seemed most resistant to the prior. It therefore appears that a Gaussian prior is most useful for unlexicalised models, and that for models built from complex, overlapping features, other forms of smoothing must be used instead.

7 Comments

We argued that RFM estimation for broad-coverage attribute-valued grammars could be made computationally tractable by training upon an informative sample. Our small-scale experiments suggested that using those parses that could be efficiently unpacked (*SCFG* sampling) was almost as effective as sampling from all possible parses (*Rand* sampling). Also, we saw that models should not be both built and also estimated using all possible parses. Better results can be obtained when models are built and trained using an informative sample.

Given the relationship between sample size and model complexity, we see that when there is a danger of overfitting, one should build models on the basis of an informative set. However, this leaves open the possibility of training such a model upon a superset of the informative set. Although we have not tested this scenario, we believe that this would lead to better results than those achieved here.

The larger scale experiments showed that RFMs can be estimated using relatively long sentences. They also showed that a simple Gaussian prior could reduce the effects of overfitting. However, they also showed that excessive overfitting probably required an alternative smoothing approach.

The smaller and larger experiments can be both viewed as (complementary) ways of dealing with overfitting. We conjecture that of the two approaches, the informative sample approach is preferable as it deals with overfitting directly: overfitting results from fitting to complex a model with too little data.

Our ongoing research will concentrate upon stronger ways of dealing with overfitting in lexicalised RFMs. One line we are pursuing is to combine a compression-based prior with an exponential model. This blends MDL with Maximum Entropy.

We are also looking at alternative template sets. For example, we would probably benefit from using templates that capture more of the syntactic context of a rule instantiation.

Acknowledgments

We would like to thank Rob Malouf, Donnla Nic Gearailt and the anonymous reviewers for comments. This work was supported by the TMR Project *Learning Computational Grammars*.

References

- Steven P. Abney. 1997. Stochastic Attribute-Value Grammars. *Computational Linguistics*, 23(4):597–618, December.
- Miles Osborne 1999. DCG Induction using MDL and Parsed Corpora. In James Cussens, editor, *Learning Language in Logic*, pages 63–71, Bled, Slovenia, June.
- Ted Briscoe and John Carroll. 1996. Automatic Extraction of Subcategorization from Corpora. In *Proceedings of the 5th Conference on Applied NLP*, pages 356–363, Washington, DC.
- John Carroll and Ted Briscoe. 1992. Probabilistic Normalisation and Unpacking of Packed Parse Forests for Unification-based Grammars. In *Proceedings of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, pages 33–38, Cambridge, MA.
- Stanley Chen and Ronald Rosenfeld. 1999a. Efficient Sampling and Feature Selection in Whole Sentence Maximum Entropy Language Models. In *ICASSP'99*.
- Stanley F. Chen and Ronald Rosenfeld. 1999b. A Gaussian Prior for Smoothing Maximum Entropy Models. Technical Report CMU-CS-99-108, Carnegie Mellon University.
- Eirik Hektoen. 1997. Probabilistic Parse Selection Based on Semantic Cooccurrences. In *Proceedings of the 5th International Workshop on Parsing Technologies, Cambridge, Massachusetts*, pages 113–122.
- Mark Johnson, Stuart Geman, Stephen Cannon, Zhiyi Chi, and Stephan Riezler. 1999. Estimators for Stochastic “Unification-based” Grammars. In *37th Annual Meeting of the ACL*.
- J. Lafferty, S. Della Pietra, and V. Della Pietra. 1997. Inducing Features of Random Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, April.
- Jorma Rissanen. 1989. *Stochastic Complexity in Statistical Inquiry*. Series in Computer Science - Volume 15. World Scientific.

LECTURES ON THE FOUNDATIONS OF HPSG

CARL POLLARD
OHIO STATE UNIVERSITY

1 General character of HPSG

1.1 Similarities to Chomsky's EST/GB Framework

1.1.1 Same principal goal: To characterize human linguistic competence

i.e., to construct a scientific theory of the system of knowledge that is embodied in the human mind/brain which makes language possible. THIS IS DISTINCT FROM CONSTRUCTING A PSYCHOLINGUISTIC THEORY (a theory of how the mind/brain USES that knowledge to produce and interpret utterances). Knowledge of language is like a database or a system of knowledge representation; it is not a set of algorithms or procedures for processing language (though such processes must have access to the system of linguistic knowledge: see below on “psycholinguistic responsibility.”). Part of the goal of characterizing competence is determining what all languages have in common (so-called “Universal Grammar”).¹

1.1.2 Same empirical base: acceptability judgements of speakers

However, care is taken to consider judgements only relative to specified classes of contexts. It is important to control for differing abilities of native speakers to imagine contexts in which a given string becomes acceptable.

1.1.3 Multiple representations

more or less analogous to “levels” of representation in EST/GB. But these are simultaneous (or coexisting, or parallel) structures that are mutually constrained by the grammar; all are parts of a single larger structures and none is sequentially derived from another.

1.1.4 Grammaticality is determined by the interaction between the lexicon and general well-formedness principles

(roughly as in GB) rather than by large numbers of construction-specific rules (as in Chomsky's “standard theory”, classical Montague grammar, or GPSG).

1.1.5 Many EST/GB concepts have HPSG analogs.

Examples include θ -roles, indices, agreement features, and traces. But in HPSG these are all carefully formalized so that empirically vulnerable predictions can be made.

¹Transformational grammar has always made a point of being concerned with limiting the range of possible variation across languages and explaining how language can be acquired. HPSG researchers consider these to be interesting long-term goals, but don't think enough is known yet to warrant positing any empirical hypotheses.

1.1.6 Likewise many EST/GB principles have rough HPSG analogs.

Examples include binding principles A, B, and C, and constraints on “extraction” (such as subadjacency and ECP).

1.2 Sociological differences

1.2.1 No “fearless leader”

There is no Chomsky-like figure who is always assumed to be basically on the right track no matter what s/he proposes. HPSG research is normal science: the testing of hypotheses that appear plausible given accepted assumptions. The goal is not to fill in the details of a vague theory which is assumed to be basically right, but to successively replace empirical hypotheses with ones that make better predictions.

1.2.2 Receptivity to adjacent technology

HPSG was developed in an environment where some familiarity with such technical bodies of knowledge as logic, set theory, algebra, graph theory and/or theoretical computer science could be assumed. Practitioners are willing to acquire a wide range of technical tools, and apply them in the interest of making their analyses and theories clearer and more precise. Consequently, they can have more confidence in their proposals.

1.3 Methodological differences

1.3.1 Freedom from typological bias

It is not assumed that all languages are basically like English. Languages are not assumed to vary without limit, but the starting assumption is that we do not know what the range of variation is, and we can take very little for granted when we begin to try to uncover the structure of a language.

1.3.2 Nonprimacy of syntax

HPSG is not syntactocentric. There is no assumption that syntax is somehow primary, and that morphology is done “in the syntax” (cf. affix hopping, head movement). Phonology and semantics are not “interpretations” of syntactic structures as in transformational models from *Syntactic Structures* to the Minimalist Program. Instead, it is assumed that we need to understand several different systems of linguistic knowledge, including syntax, semantics, morphology, phonology/prosody, and pragmatics/discourse. In HPSG theories, most grammatical principles don’t involve just one of these, but instead constrain the relationship between two or more.

1.3.3 Empirical adequacy

HPSG is done bottom-up or inductively, generalizing from specifics instead of starting with grand generalities and looking for particulars that confirm them. Thus it employs the fragment methodology: make a precise, falsifiable hypothesis that accounts for a wide range of facts (i.e. get the details right for a subpart of the language), then revise the hypothesis to expand coverage. Precise, empirically vulnerable generalizations with broad, determinable consequences (which might not follow from very deep principles) are valued over deep principles that are so vague that their empirical consequences cannot be deduced.

1.3.4 Psycholinguistic responsibility

Grammars, as models of competence, should nonetheless be capable in principle of being exploited by plausible psycholinguistic models. Thus, they should not be based IRREDUCIBLY on computations that a language user would be incapable of carrying out. And thus there is a somewhat different emphasis in the subject matter: instead of a (largely rhetorical) concern with how linguistic knowledge is acquired, rather one is concerned with how it is organized so as to make language use possible. The language user is viewed more as an information-processing agent than as language-acquisition device. Ultimately, grammatical knowledge must be organized in such a way that it can be employed efficiently in a wide variety of tasks, including production, understanding, translation, language games, making acceptability judgements, and more. But the linguistic framework itself should be TASK-NEUTRAL.

1.3.5 Generativity

HPSG research takes it for granted (as Chomsky originally did) that the goal of a grammatical theory of language X is minimally to tell what the well-formed structures of language X are (“descriptive adequacy”). Thus, formal precision is necessary (the current Chomskyan view is that it is “premature” and therefore undesirable). In an adequate formalization, according to Pullum (1989):

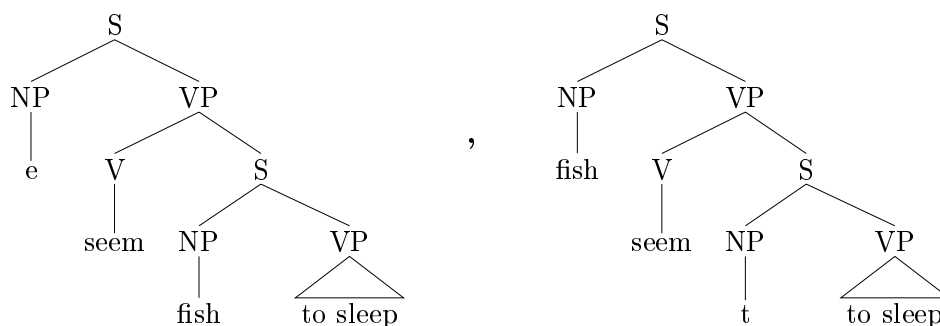
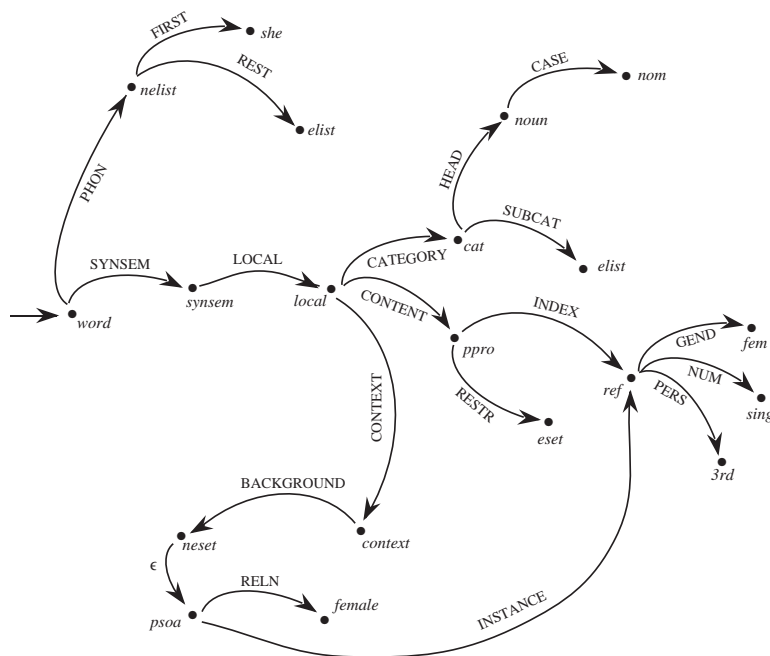
1. It must be made clear what mathematical structures are used to model different kinds of linguistic entities. (In HPSG, the mathematical structures are graphs of a certain kind called feature structures.)
2. It must be determinate what the actual assertions of the theory are. They don’t have to be framed in formal logic or Prolog or Lisp or C++ though sometimes it is helpful to do this. Careful English (or Korean, etc.) will do, as long as it is clear, specific, and unambiguous.
3. Given a grammar G and a mathematical object O used as a model of a candidate linguistic entity (a representation), there has to be a way to tell whether or not O satisfies the constraints imposed by G.

1.4 Architectural differences

There are large-scale differences in how theories are formulated or organized.

1.4.1 Structures employed as mathematical idealizations of linguistic entities

HPSG employs different mathematical structures from EST/GB. The mathematical structures of transformational theories are sequences of phrase-markers (trees, i.e., rooted, directed, graphs satisfying the single mother condition, with nodes labelled by category symbols).



In HPSG, the mathematical structures are feature structures: rooted, connected, directed graphs, with each arc labelled by a feature name, and each node labelled by a species name (the name of a kind of linguistic object).

- A **rooted** graph is one that has a distinguished node called the **root**.
- In a **connected** graph, every node is reachable from the root by a finite sequence of edges.
- In a **directed** graph, the edges are directed (i.e. they are ordered pairs of nodes, not just doubleton sets of nodes).

1.4.2 Grammars are formalized

Fully formalized, an HPSG grammar is formulated as a set of well-formedness constraints on feature structures, of which each constraint is a (nonlogical) axiom in a certain kind of formal language called a feature logic. (In these notes, we use a slight notational variant of King's (1989) SRL). The familiar attribute-value matrices (AVMs) are an INFORMAL substitute for feature logic constraints. Feature structures and feature logic satisfy the criteria of formal precision stated above in subsection 1.3.5. The constraints tell which feature structures are well-formed representations of linguistic

entities. Technically, a grammar is a logical theory, and a well-formed structure is a model of the theory.

The feature logic is itself unconstrained in terms of what constraints it can express. It is like predicate calculus or LISP in this respect: the constraints don't come from limits on what theories the logic can express; they ARE a theory that the logic expresses. Thus HPSG employs an EXPRESSIVE formalism, not a constrained formalism.

1.4.3 Nonderivationality

HPSG is nonderivational. It employs PARALLEL REPRESENTATIONS which are MUTUALLY CONSTRAINED by the grammar. This is also true of other nontransformational frameworks, such as LFG, APG, autolexical syntax, construction grammar, and Jackendoff's (1996) programmatic Representational Modularity framework.

HPSG employs no transformations or other destructive operations to sequentially derive one structure (or representation) from another. Instead, the different representations (or levels, in Ladusaw's (1988) terminology) are just subparts (features, actually) of a single larger structure, and they are related not by destructive operations, but rather by declarative (asserted) constraints of the grammar.² (Thus, the substructure reached from the root by following the PHON path is a rough correspondent of the GB level PF; and the one reached by the path SYNSEM|LOC|CAT in a verb corresponds roughly to the DS of the sentence headed by that verb; the substructure reached by the path SYNSEM|LOC|CONT corresponds roughly to the GB level LF. Understood this way, HPSG has considerably more levels than GB.)

1.4.4 Structural uniformity

HPSG is "fractal" (structurally uniform as the parts get smaller). Every sign down to the word level (not just the root clause) has features corresponding (inter alia) to phonetic, syntactic, and semantic aspects of linguistic structures

1.4.5 No lexical insertion

Words are just the same kind of thing (namely, signs) as phrases; they are just small-scale constrained parallel structures. Thus, there is no distinction between terminal nodes and preterminals, and no issue (as in recent Minimalist Program work) of whether lexical insertion is "early" or "late". This is a consequence of the theory being nonderivational and fractal.

1.4.6 Locality

HPSG employs only local constraints. There are no global constraints constraining structures relative to other structures. Well-formedness is determined completely with reference to a given structure, and not via comparison with any other "competing" structure (as it might be in MP and OT).

²IRREDUCIBLE use of transformations is hard to reconcile with psycholinguistic responsibility: to determine PF from LF, an SS must first be determined, from which one could work back to a DS source, and ahead to derive a PF. The first two steps require "inverting transformations," which is a computational nightmare. Similarly for determining LF from PF. Although some GBists (e.g. Koster (1987), Brody (1995)) have argued that GB theory could just as well be reformulated without transformations, Chomsky has never been sympathetic to this position, and with the Minimalist Program, it is not really an option.

1.5 Technical Differences

There are also smaller-scale differences in mechanisms employed by the framework. Some representative examples:

- Tree-configurational notions like c-command and government have no role in HPSG, which employs instead the traditional relation of **obliqueness of grammatical relations**, where obliqueness increases from *Subject* to *Primary Object* to *Secondary Object* to *Oblique PPs* and *VP, AP, and S Complements*. Thus, e.g. Principle A of the binding theory requires not that an r-pronoun (anaphor) be coindexed with a c-commanding NP in some local domain, but rather that it be coindexed with a less oblique argument of the same head (if there is one).
- **Nonconfigurational definitions of grammatical relations.** Grammatical relations are defined NONconfigurally, in terms of their relative order on the lists that are the values of the VALENCE features. Thus, the subject is the unique item on the SUBJ list; the primary object is the first item on the COMPS list, etc., rather than being sister-of-VP-and-daughter-of-S, or daughter-of-VP-and-closest-sister-of-V, as in transformational theories from *Aspects* through GB.
- **Selection of subjects and specifiers.** Subjects and specifiers (and their detailed properties, such as number and definiteness) can be selected (“subcategorized for”) by lexical heads in HPSG, just as complements are. Thus subjectless sentences are just as possible as intransitive verbs, since a verb can select for “no subject”.
- **Semantic roles (“ θ -roles”)** are assigned to subjects directly, in the same way as to complements, not indirectly.
- Likewise, **either subject or object can fail to be assigned a semantic role.** Among other things, this means that “raising-to-object” analyses are not ruled out. In GB, by contrast, nonthematic objects are impossible.
- There is **no requirement that every VP have a subject**; thus, verbs that appear to take infinitive VP complements can be treated as actually doing so, with no need for a phonologically null subject (PRO or NP-trace). Control is treated as coindexing of the controller phrase with the value of the SUBJ valence feature of the VP complement.

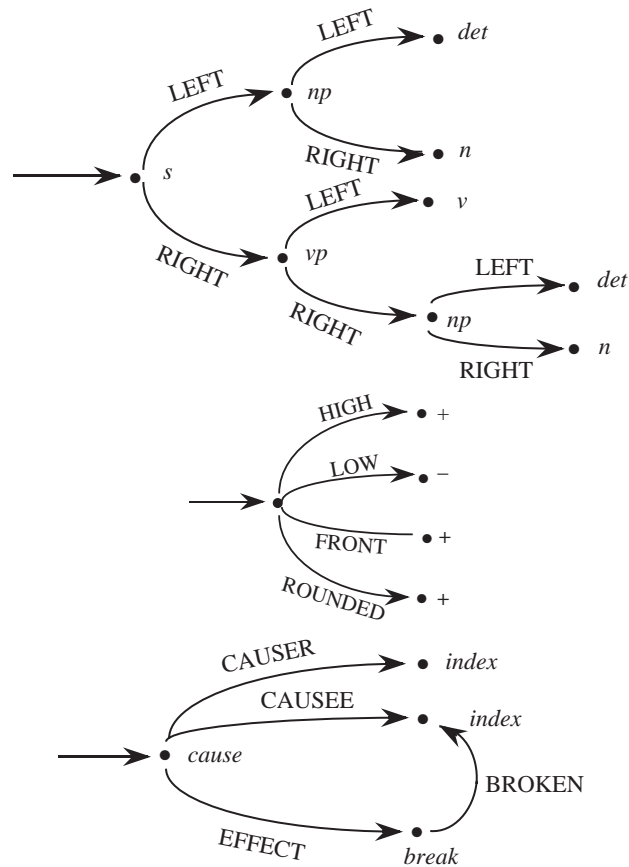
The following differences are directly related to the nonderivationality of HPSG.

- **No NP-movement** (and, more generally, no movement at all.)

Passive: Passive verbs are in the lexicon (perhaps mostly derived by lexical rule). They project a phrase according to the same constraints as active verbs do.

Raising: STRUCTURE-SHARING between the NP subject or object and the member of the SUBJ valence feature list of the VP complement takes the place of NP-movement. Thus, the analysis of raising structures differs from the analysis of equi-structures only in that 1) raising predicates systematically have one semantic role fewer than equi predicates with the same number of syntactic arguments, and 2) raising verb select (as subject or object) for the same structure that the VP complement selects for its subject, while in the case of an equi verb, the subject or object is only coindexed with the complement subject. Moreover, raising to subject and raising to object are treated on a par.

- **No WH-movement:** The value of the SLASH feature throughout the path from the verb whose argument is “missing” to the filler WH-element is structure-shared with (1) the LOCAL value of the filler element, and (2) the LOCAL value of the missing argument.
- **No head movement:**
 - Verb inflection:** finite verbs are specified in the lexicon to select subjects bearing specific agreement features. Thus there is no movement between V and Infl (in fact there is no need for a distinct Infl node).
 - Inversion:** Inverted structures are “flat” structures, not derived from uninverted structures. Thus there is no movement from Infl to Comp.
- More generally, there are **no null functional heads** (T, AgrO, AgrS, etc.). Instead the corresponding work in HPSG is done by features.
- Likewise, there are **no null complementizers**.



2 Foundations of the HPSG Formalism

2.1 Feature structures

Feature structures are mathematical objects—graphs of a certain kind—that are used as theoretical models of structured entities of all sorts in the real world, not just linguistic entities. Thus, the diagrams above represent a high front rounded object, a constituent structure diagram, and the relation of breaking. (Note: these are for illustrative purposes only. They are not being posited as serious linguistic analyses.)

3

2.2 Formal Machinery

Assume that there are given, disjoint, finite sets \mathcal{F} of *feature names*, and \mathcal{S} of *species names*.

Definition: A **feature graph** is an ordered triple $\mathcal{G} = \langle \mathcal{U}, \mathcal{S}, \mathcal{F} \rangle$, where:

- \mathcal{U} is a set (called the *nodes* of \mathcal{G});
- \mathcal{F} is a function which associates with each feature name f a partial function $F(f)$ (also written $f_{\mathcal{G}}$) from \mathcal{U} to \mathcal{U} ; and
- \mathcal{S} is a function which associates with each species name s a subset of \mathcal{U} ; $\mathcal{S}(s)$ is also written $s_{\mathcal{G}}$).

Mathematically, a feature graph is just a sorted unary partial algebra, where S interprets the sorts (species names) and F interprets the operations (feature names).

There is also a more primitive notion of (directed) graph, where all we have is nodes and (directed) edges (with no species or features). This is defined as follows:

Definition: A **(directed) graph** is an ordered pair $\mathcal{G} = \langle \mathcal{U}, \rightarrow \rangle$, where:

- a. \mathcal{U} is a set (called the *nodes* of \mathcal{G});
- b. \rightarrow is a binary relation on \mathcal{U} (i.e., a subset of $\mathcal{U} \times \mathcal{U}$) called the *edges* of \mathcal{G} . (We write $p \rightarrow q$ to mean $\langle p, q \rangle \in \rightarrow$.)

Often in working with graphs we single out a node which serves as a sort of “home base”. Technically:

Definition: A **pointed graph** is an ordered triple $\mathcal{G} = \langle q_0, \mathcal{U}, \rightarrow \rangle$, where $\langle \mathcal{U}, \rightarrow \rangle$ is a graph (called the *underlying* graph of \mathcal{G}) and $q_0 \in \mathcal{U}$; q_0 is called the *point* of \mathcal{G} .

Definition: Given a graph $\langle \mathcal{U}, \rightarrow \rangle$ and two nodes $p, q \in \mathcal{U}$, we say q is **accessible** from p just in case $p \rightarrow^* q$. (For any binary relation R , R^* is the reflexive transitive closure of R .)

Informally, q is accessible from p just in case there is a path (finite sequence of edges) from p to q .)

Definition: A pointed graph is called **accessible** (or an **apg**) iff every node is accessible from the point. In this case, the point is called the **root** of the apg.

Thus, if $\mathcal{G} = \langle q_0, \mathcal{U}, \rightarrow \rangle$ is an apg, then q_0 is the root and every node in \mathcal{U} is accessible from q_0 . Trees are a familiar kind of apg. (An apg is a tree just in case each node is accessible from the root via a unique path.)

Observation: Let $\mathcal{G} = \langle \mathcal{U}, S, F \rangle$ be a feature graph, and let $\rightarrow_F \subseteq \mathcal{U} \times \mathcal{U}$ be defined as $\bigcup \{f_G : f \in \mathcal{F}\}$. Then $\langle \mathcal{U}, \rightarrow_F \rangle$ is a graph.

That is, given a feature graph, we can form the graph whose set of edges is the union of the interpretations of the feature names. Intuitively, this amounts to throwing away all the labels on the nodes and edges.

Definition: A **feature structure** is a quadruple $\langle q_0, \mathcal{U}, S, F \rangle$, where $\langle \mathcal{U}, S, F \rangle$ is a feature graph, and $\langle q_0, \mathcal{U}, \rightarrow_F \rangle$ is an apg. We call q_0 the **root** of the feature structure.

Feature structures are what we will use as our mathematical idealizations of linguistic entities (i.e. analyses of expressions).

2.3 Graph Notation

Nodes are represented by dots, and edges (or arcs) by arrows. Thus, in the graph $\bullet \longrightarrow \bullet$, if the left dot is p and the right dot is q , then $p \rightarrow q$. For a feature graph, an arrow labelled with a feature name f means that $f_G(p) = q$, where p is represented by the dot from which the arrow originates and q is represented by the dot the arrow points to. The root of a feature structure is indicated by a short boldface arrow to a node: $\rightarrow \bullet$.

2.4 Feature structures generated by nodes in feature graphs

Definition: Let $\mathcal{G} = \langle \mathcal{U}, \mathcal{S}, \mathcal{F} \rangle$ be a feature graph and $q \in \mathcal{U}$ a node. The feature structure **generated by q in \mathcal{G}** , \mathcal{G}_q , is $\langle q, \mathcal{U}_q, \mathcal{S}_q, \mathcal{F}_q \rangle$, where:

- a. $\mathcal{U}_q = \{p \in \mathcal{U} \mid q \rightarrow_{\mathcal{F}}^* p\}$;
- b. for each s in \mathcal{S} , $\mathcal{S}_q(s) = s_{\mathcal{G}} \cap \mathcal{U}_q$;
- c. for each f in \mathcal{F} , $\mathcal{F}_q(f) = f_{\mathcal{G}} \cap (\mathcal{U}_q \times \mathcal{U}_q)$.

Thus the feature structure generated by a node q in a feature graph \mathcal{G} contains only the nodes of \mathcal{G} accessible from q , and the interpretations of species names and feature names are obtained by restricting their interpretations in \mathcal{G} in the obvious way.

2.5 Feature structure homomorphisms and isomorphisms

Let $\mathcal{G} = \langle q_0, \mathcal{U}, \mathcal{S}, \mathcal{F} \rangle$ and $\mathcal{G}' = \langle q'_0, \mathcal{U}', \mathcal{S}', \mathcal{F}' \rangle$.

Definition: A **homomorphism** from \mathcal{G} to \mathcal{G}' is a function $\eta : \mathcal{U} \rightarrow \mathcal{U}'$ such that

- a. $\eta(q_0) = q'_0$;
- b. for all $q \in \mathcal{U}$ and $f \in \mathcal{F}$, if $f_{\mathcal{G}}$ is defined at q , then $f_{\mathcal{G}'}$ is defined at $\eta(q)$, and $f_{\mathcal{G}'}(\eta(q)) = \eta(f_{\mathcal{G}}(q))$;
- c. for all $q \in \mathcal{U}$ and $s \in \mathcal{S}$, if $q \in s_{\mathcal{G}}$, then $\eta(q) \in s_{\mathcal{G}'}$.

This is just like a homomorphism in algebra (thinking of the interpretations of the feature names as the algebra operations), with the additional provisos that the species and the root must be preserved.

Definition: A homomorphism is called an **isomorphism** if it has an inverse which is also a homomorphism. It follows that isomorphisms are both one-to-one and onto.

Thus isomorphic feature structures “look the same”; they just have different nodes. It is easy to see that, for any feature structure $\mathcal{G} = \langle q_0, \mathcal{U}, \mathcal{S}, \mathcal{F} \rangle$, the identity function on \mathcal{U} , $id_{\mathcal{U}}$, is an isomorphism from \mathcal{G} to \mathcal{G} . Also, if η is a homomorphism from \mathcal{G} to \mathcal{G}' and η' is a homomorphism from \mathcal{G}' to \mathcal{G}'' , then $\eta' \circ \eta$ is a homomorphism from \mathcal{G} to \mathcal{G}'' .

In HPSG, as explained below, a grammar is a kind of logical theory whose intended interpretations are feature structures. Thus the grammar picks out a certain class of feature structures, namely the ones that are models of the grammar. Now clearly, for any feature structure, there are infinitely many other feature structures (in fact, a proper class of them) isomorphic to it. However, as linguists, when we are given a grammar, we don't really care about the whole class of models. All we really care about is which feature structures are models *up to isomorphism*; e.g. the fact that there is an infinity of different feature structure models of the sentence *poor John ran away* is not of interest to us, since all these models are isomorphic. What we would really like is a way to pick out from each isomorphism class of models a unique canonical representative. Then we could consider the grammar to **generate** precisely the set of feature structures which are the canonical representatives of the various isomorphism classes of models. This idea connects the model-theoretic interpretation of grammars with the standard linguistic notion of (strong) generative capacity.

In fact, there is a rather simple mathematical technique for constructing such representative feature structures; roughly speaking, the trick is to choose feature structures whose nodes are equivalence classes of paths (strings of feature names). The precise construction, which is a slight variant of Moshier's (1988) notion of an *abstract* feature structure, is described in Pollard 1998.

2.6 Paths

A **path** is a member of \mathcal{F}^* , i.e., it is a finite sequence (string) of feature names. As usual, the null path (path of length zero) is denoted by ϵ . Given a feature graph $\mathcal{G} = \langle \mathcal{U}, \mathcal{S}, \mathcal{F} \rangle$, the function $F : \mathcal{F} \rightarrow [\mathcal{U} \rightarrow \mathcal{U}]$ can be extended to a function $\tilde{F} : \mathcal{F}^* \rightarrow [\mathcal{U} \rightarrow \mathcal{U}]$ as follows: \tilde{F} is defined by recursion on path lengths n such that:

- a. if a path π is of length 0 (i.e., $\pi = \epsilon$) then $\tilde{F} = id_{\mathcal{U}}$;
- b. if π is of length $n > 0$ and $\pi = f\pi'$ where $f \in \mathcal{F}$ and π' is a path of length $n-1$, then

$$\tilde{F}(\pi) = (\tilde{F}(\pi')) \circ (F(f)).$$

(That is, for a path of length $n > 0$, the function it denotes is the composition of two functions: the one denoted by its final subpath of length $n-1$, and the one denoted by its initial feature.)

$\tilde{F}(\pi)$ is also written $\pi_{\mathcal{G}}$. Thus

$$\epsilon_{\mathcal{G}} = id_{\mathcal{U}}$$

and if $\pi = f\pi'$, then

$$\pi_{\mathcal{G}} = \pi'_{\mathcal{G}} \circ f_{\mathcal{G}}$$

Also, $(f_1 \cdots f_n)_{\mathcal{G}} = (f_n)_{\mathcal{G}} \circ \cdots \circ (f_1)_{\mathcal{G}}$.

2.7 King formulas

Given disjoint finite sets \mathcal{F} (feature names) and \mathcal{S} (species names), the set \mathcal{K} of **King formulas** (cf. King (1989)) is defined recursively as follows:

- a. for each s in \mathcal{S} , $s \in \mathcal{K}$;
- b. $\top \in \mathcal{K}$;
- c. for each path π and each formula $\phi \in \mathcal{K}$, $(\pi : \phi) \in \mathcal{K}$
- d. for π_1 and π_2 paths $(\pi_1 \doteq \pi_2) \in \mathcal{K}$;
- e. for π_1 and π_2 paths $(\pi_1 \neq \pi_2) \in \mathcal{K}$;
- f. for ϕ and $\psi \in \mathcal{K}$, $(\phi \wedge \psi) \in \mathcal{K}$;
- g. for ϕ and $\psi \in \mathcal{K}$, $(\phi \vee \psi) \in \mathcal{K}$;
- h. for ϕ and $\psi \in \mathcal{K}$, $(\phi \rightarrow \psi) \in \mathcal{K}$;
- i. for ϕ and $\psi \in \mathcal{K}$, $(\neg\phi) \in \mathcal{K}$;
- j. nothing else is in \mathcal{K} .

As described below, King formulas are used in HPSG both as descriptions of feature structures and as constraints in grammars. To explain the difference, we begin by defining a **satisfaction** relation \models between feature structures and King formulas, analogous to the relation between first-order models and first-order formulas.

2.8 Satisfaction

The satisfaction relation \models between a feature structure $\mathcal{G} = \langle q_0, \mathcal{U}, S, F \rangle$ and a King formula is defined by structural recursion on formulas as follows:

- a. $\mathcal{G} \models s$ iff $q_0 \in s_{\mathcal{G}}$;
- b. $\mathcal{G} \models \top$ never;
- c. $\mathcal{G} \models (\pi : \phi)$ iff $\pi_{\mathcal{G}}$ is defined at q_0 and $\mathcal{G}_{\pi_{\mathcal{G}}(q_0)} \models \phi$;
- d. $\mathcal{G} \models (\pi_1 \doteq \pi_2)$ iff $(\pi_1)_{\mathcal{G}}$ and $(\pi_2)_{\mathcal{G}}$ are both defined at q_0 and have the same values;
- e. $\mathcal{G} \models (\pi_1 \neq \pi_2)$ iff $(\pi_1)_{\mathcal{G}}$ and $(\pi_2)_{\mathcal{G}}$ are both defined at q_0 and have different values;
- f. $\mathcal{G} \models (\phi \wedge \psi)$ iff $\mathcal{G} \models \phi$ and $\mathcal{G} \models \psi$;
- g. $\mathcal{G} \models (\phi \vee \psi)$ iff $\mathcal{G} \models \phi$ or $\mathcal{G} \models \psi$;
- h. $\mathcal{G} \models (\phi \rightarrow \psi)$ iff $\mathcal{G} \not\models \phi$ or $\mathcal{G} \models \psi$;
- i. $\mathcal{G} \models (\neg\phi)$ iff $\mathcal{G} \not\models \phi$.

Note: if \mathcal{G} and \mathcal{G}' are isomorphic feature structures, ϕ is a King formula, and $\mathcal{G} \models \phi$, then also $\mathcal{G}' \models \phi$.

If $\mathcal{G} \models \phi$, then we say \mathcal{G} **satisfies** ϕ or ϕ **describes** \mathcal{G} . A formula is called **satisfiable** if some feature structure satisfies it (or, equivalently, if it describes some feature structure). For a set of formulas Φ , we say $\mathcal{G} \models \Phi$ iff $\mathcal{G} \models \phi$ for every $\phi \in \Phi$.

2.9 Entailment and (semantic) equivalence

Let Φ be a set of formulas and ϕ a formula. Then we say Φ **entails** ϕ , written $\Phi \models \phi$, provided for all feature structures \mathcal{G} , if $\mathcal{G} \models \Phi$ then $\mathcal{G} \models \phi$. If Φ contains only one formula ϕ , we usually write $\phi \models \psi$ for $\Phi \models \psi$. If ϕ and ψ are two formulas such that $\phi \models \psi$ and $\psi \models \phi$, then we say ϕ and ψ are (semantically) equivalent, written $\phi \approx \psi$. Equivalent formulas describe the same feature structures.

2.10 Constraints and Grammars

Let ϕ be a formula and \mathcal{G} a feature structure (or more generally, a feature graph). Then we say \mathcal{G} **models** ϕ (or, is a **model** of ϕ) provided, for every node q of \mathcal{G} , $\mathcal{G}_q \models \phi$. For Φ a set of formulas (e.g., a theory), we say \mathcal{G} **models** Φ if \mathcal{G} models ϕ for every $\phi \in \Phi$. Note: if \mathcal{G} models ϕ , then $\mathcal{G} \models \phi$, but the converse is in general false.

It is important to understand the relation between a *description* and a *constraint*. A formula is called a **description** when we are concerned only with which feature structures satisfy it. By contrast, a formula is called a **constraint** when we are concerned with which feature structures model it.

A **grammar** is just a set of formulas viewed as constraints. If G is a grammar and \mathcal{G} a feature structure which models G , we also say G **generates** \mathcal{G} , or \mathcal{G} is **well-formed** relative to G .³ We can now consider some examples of constraints typically employed in HPSG grammars.

³Strictly speaking, the grammar generates only the models which are canonical representatives of isomorphism classes of models. See Pollard 1998.

2.11 Species disjointness

It is standard to assume that, in order to be well-formed, a feature structure must be such that each of its nodes belongs to only one species. We can enforce this condition by including in the grammar, for each pair of distinct species names s and s' the constraint $\neg(s \wedge s')$.

2.12 Closed World Assumption

Another standard assumption in HPSG is that there aren't any linguistic entities beyond the species corresponding to the species names in \mathcal{S} . This can be thought of as a version of the fragment methodology. Another way to say this is that in order for a feature structure to be well-formed, every one of its nodes must belong to at least one (and given species disjointness, therefore exactly one) of the species. This can be expressed as the constraint:

$$\bigvee_{i=1}^n s_i \text{ (where } \mathcal{S} = \{s_1, \dots, s_n\}\text{)}$$

(Note: for ϕ_1, \dots, ϕ_n formulas, the notation $\bigvee_{i=1}^n \phi_i$ is a shorthand interpreted as follows:

for $n=1$, $\bigvee_{i=1}^n \phi_i$ means ϕ_1

for $n > 1$, it means $(\bigvee_{i=1}^{n-1} \phi_i) \vee \phi_n$

Thus, e.g., if $n = 3$, $\bigvee_{i=1}^n \phi_i$ is shorthand for $((\phi_1 \vee \phi_2) \vee \phi_3)$. A similar convention applies for $\bigwedge_{i=1}^n \phi_i$.)

2.13 Feature Geometry Constraints

These are constraints which tell us what features different species have, and what species the values of those features belong to. In HPSG it is standardly assumed that for each feature, there is a set of species such that that feature is defined for all nodes belonging to that species, but undefined for all other nodes. Thus, for each feature f , there is a constraint of the form

$$((f \doteq f) \leftrightarrow \bigvee_{i=1}^m t_i)$$

where each of the t_i is a species. For example, in HPSG, words and phrases (but nothing else) have the feature PHONOLOGY. Thus HPSG grammars contain the constraint⁴

$$((\text{PHONOLOGY} \doteq \text{PHONOLOGY}) \leftrightarrow (\text{word} \vee \text{phrase}))$$

(Thus, anything that has a PHONOLOGY attribute is either a word or a phrase, and every word or phrase has a PHONOLOGY attribute.)

Another type of feature geometry constraint tells, for a given species and a given feature defined for that species, what species the value of that feature can be. Such constraints have the form

⁴As in propositional logic, " $\phi \leftrightarrow \psi$ " is shorthand for $((\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi))$.

$$(s \rightarrow (f : \bigvee_{i=1}^m t_i))$$

For example, a German HPSG grammar might include the constraint

$$(noun \rightarrow (CASE : (nom \vee acc \vee gen \vee dat))).$$

2.14 Other constraints

Besides the feature geometry constraints, species disjointness, and the closed world assumption (which are sometimes referred to collectively as *ontological* constraints), grammars must contain further constraints which serve to determine (among those feature structures which satisfy the ontological constraints) which feature structures are well-formed. These constraints do the work done in other frameworks by such devices as lexical entries, phrase-structure (or immediate-dominance) rules, and principles of well-formedness (e.g., Subjacency, Binding Principle A, etc.). For example, it is standardly assumed in HPSG that for all headed phrases, the “head features” are the same as on the head daughter. This is formally expressed by the constraint

$$(headed-phrase \rightarrow (SYNSEM|LOC|CAT|HEAD \doteq \\ HEAD-DTR|SYNSEM|XSLOC|CAT|HEAD))$$

In HPSG, the lexicon is also a constraint. It has the form

$$(word \rightarrow \bigvee_{i=1}^N \lambda_i)$$

where each of the λ_i is a lexical entry. That is, each lexical entry is a formula (or description) which presents one of the (finitely many) options for a feature structure of species *word* to be a well-formed (structural representation of a) word. The lexicon, as a constraint, simply requires that every word take one of these options.

Similarly, there is a constraint on phrases of the form

$$(phrase \rightarrow \bigvee_{i=1}^M \rho_i)$$

where each of the ρ_i is an immediate dominance schema (or phrase-type). That is, each of the ρ_i presents one of the (small number of) options for a feature structure of species *phrase* to be a well-formed (structural representation of a) phrase. Thus the ρ_i do the same kind of work in HPSG as schematic immediate dominance rules (such as the rules of \bar{X} theory) do in GB theory.

2.15 Formal properties of King formulas and grammars

Here we assemble some basic facts about formal properties of King formulas and grammars. For this discussion, assume we are given a set \mathcal{O} of ontological constraints as discussed in 11-13 above. A feature structure is called **ontologically acceptable** if it satisfies all the ontological constraints (species disjointness, closed world, and feature geometry).

- a. There is a proof theory for \mathcal{K} which is sound and complete relative to ontological acceptability. Among other things, this means that for any two formulas ϕ and ψ , $\mathcal{O} \vdash (\phi \rightarrow \psi)$ iff every ontologically acceptable feature structure that satisfies ϕ also satisfies ψ .

- b. Given a formula ϕ , it is decidable whether or not ϕ is satisfied by some ontologically acceptable feature structure or other. Likewise (since negation is classical), it is decidable whether ϕ is valid relative to ontological acceptability (i.e., whether EVERY ontologically acceptable feature structure satisfies ϕ).
- c. Given a grammar G and a formula ϕ it is in general UNdecidable whether G predicts a feature structure which satisfies ϕ (or equivalently, whether there is a feature structure which models G and satisfies ϕ).
- d. Given a set of ontological constraints \mathcal{O} and a formula ϕ , there is a formula ϕ' which is (semantically) equivalent to ϕ relative to \mathcal{O} (i.e., ϕ and ϕ' are satisfied by the same ontologically acceptable feature structures) such that ϕ' is a disjunction of conjunctions of formulas of one of the three forms $(\pi : s)$, $(\pi_1 \doteq \pi_2)$, and $(\pi_1 \neq \pi_2)$. Thus, when \mathcal{K} is being used only for description (as opposed to for constraints), no descriptive power is sacrificed by abstaining from the use of \neg and \rightarrow .

2.16 Attribute-value matrices (AVMs)

King formulas are useful when great formal precision is needed (e.g., in proofs or in designing computer implementations). But they are too long and too hard to understand quickly for everyday linguistic analysis. For this reason, HPSG linguists usually use a different kind of description, attribute-value matrices (AVMs), in normal practice. The general format of an AVM is as follows:

- a. a **tag** (boxed numeral like \boxed{n}) is an AVM.
- b. a species symbol (e.g., *word*) or a disjunction of species symbols (e.g., *word* \vee *phrase*) is an AVM; this kind of AVM is called a **sort description**. (Cf. section 19 below.)
- c. a **bracketed description**, including the empty description ‘ $[\]$ ’, is an AVM.

This has the form
$$\left[\begin{array}{ll} \text{PATH}_1 & \text{AVM}_1 \\ \vdots & \\ \text{PATH}_n & \text{AVM}_n \end{array} \right]$$

In any AVM, any non-null combination of (a), (b), and (c) can be present. Thus, the following possibilities are available:

- a. just a tag: \boxed{n} ;
- b. just a sort description: s , or $s_1 \vee \dots \vee s_n$;
- c. just a bracketed description;
- d. both (a) and (b), e.g., $\boxed{n}s$;
- e. both (a) and (c), e.g., $\boxed{n} \left[\begin{array}{l} \dots \\ \dots \end{array} \right]$

f. both (b) and (c), e.g.,

$$\begin{bmatrix} s \\ \dots \\ \dots \end{bmatrix}$$

g. all three, e.g.,

$$\boxed{n} \begin{bmatrix} s \\ \dots \\ \dots \end{bmatrix}$$

Note that tags go outside the left bracket, and sort specifications go inside the left bracket at the top (or in some work, outside the left bracket at the bottom). Sort descriptions are equivalent to the identical King formulas. The lines inside a bracketed description are interpreted as equivalent to CONJUNCTION of King formulas. Co-occurrence of two or more instances of the same tag in an AVM is equivalent to path equalities ($\pi_1 \doteq \pi_2$). For example, the King formula

$$\begin{aligned} &(((\text{SUBJ|HEAD}:((\text{CASE: } \mathit{nom}) \wedge (\text{AGR } ((\text{PER: } \mathit{3rd}) \wedge (\text{NUM: } \mathit{sing})))))) \wedge \\ &(\text{HEAD: } (\mathit{verb} \wedge (\text{VFORM: } \mathit{fin}) \wedge (\text{AUX: } -))) \wedge \\ &(\text{SUBJ|HEAD|AGR} \doteq \text{HEAD|AGR})) \end{aligned}$$

is equivalent to the AVM

$$\left[\begin{array}{l} \text{SUBJ|HEAD} \begin{bmatrix} \text{CASE} & \mathit{nom} \\ \text{AGR} & \boxed{n} \begin{bmatrix} \text{PER} & \mathit{3rd} \\ \text{NUM} & \mathit{sing} \end{bmatrix} \end{bmatrix} \\ \text{HEAD} \begin{bmatrix} \mathit{verb} \\ \text{VFORM} & \mathit{fin} \\ \text{AUX} & - \\ \text{AGR} & \boxed{n} \end{bmatrix} \end{array} \right]$$

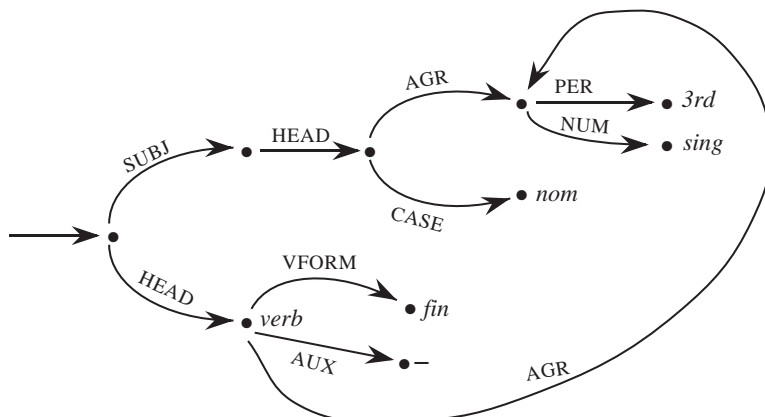
As a matter of style, when several paths share a prefix, that prefix is normally written only once for all the paths. Thus, we write:

$$\left[\text{HEAD} \begin{bmatrix} \text{VFORM} & \mathit{fin} \\ \text{AUX} & - \end{bmatrix} \right]$$

rather than:

$$\left[\begin{array}{l} \text{HEAD} \begin{bmatrix} \text{VFORM} & \mathit{fin} \end{bmatrix} \\ \text{HEAD} \begin{bmatrix} \text{AUX} & - \end{bmatrix} \end{array} \right]$$

This example makes obvious the relative advantages and disadvantages of the two styles of description. Either one is satisfied by the following feature structure:



There is no standard equivalent of path INequalities for AVMs, but one obvious way to do it is to list path inequalities as additional lines inside a bracketed description. Thus, e.g., the King formula $(\text{SUBJ|INDEX} \neq \text{OBJ|INDEX})$ would be expressed as the AVM:

$$\left[\text{SUBJ|INDEX} \neq \text{OBJ|INDEX} \right]$$

A less obvious but more readable alternative is the following:

$$\left[\begin{array}{l} \text{SUBJ|INDEX} \quad \boxed{1} \\ \text{OBJ|INDEX} \quad \boxed{2} \\ \boxed{1} \neq \boxed{2} \end{array} \right]$$

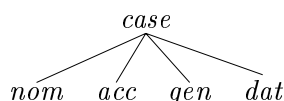
2.17 Subsumption and unification of AVMs

Given two AVMs μ_1 and μ_2 , we say that μ_1 **subsumes** μ_2 if the class of feature structures that μ_1 describes is at least as big as the class of feature structures that μ_2 describes. If ϕ_1 and ϕ_2 are King formulas equivalent to μ_1 and μ_2 respectively, then μ_1 subsumes μ_2 iff ϕ_2 entails ϕ_1 .

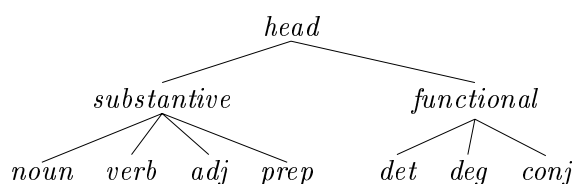
Again let μ_1 and μ_2 be two AVMs, with equivalent King formulas ϕ_1 and ϕ_2 respectively. An AVM μ_3 is called the **unification** of μ_1 and μ_2 (or more correctly, **a unification** of μ_1 and μ_2) if μ_3 is equivalent to $\phi_1 \wedge \phi_2$. Thus, μ_3 will describe those feature structures which both μ_1 and μ_2 describe (and no others).

2.18 Sorts and sort hierarchies

In writing HPSG grammars, certain disjunctions of species names are used repeatedly. For example, in a German grammar, one might often have occasion to employ the disjunction $(\text{nom} \vee \text{acc} \vee \text{gen} \vee \text{dat})$ (as in the constraint at the end of Sec. 13). Similarly, the disjunction $(\text{word} \vee \text{phrase})$ recurs frequently. Frequently used disjunctions are usually abbreviated with a single symbol, so that $(\text{word} \vee \text{phrase})$ is usually abbreviated *sign* and $(\text{nom} \vee \text{acc} \vee \text{gen} \vee \text{dat})$ is usually abbreviated *case*. Such abbreviations make grammars much easier to read and write (e.g., $((\text{PHONOLOGY} \doteq \text{PHONOLOGY}) \leftrightarrow (\text{sign}))$, $(\text{noun} \rightarrow (\text{CASE: case}))$). Of course, such abbreviations are meaningless unless they are made explicit. This is typically done by means of a diagram called a *sort hierarchy* (where a sort is just a symbol that is either a species name or an abbreviation for a disjunction of species names). Thus, the sort hierarchy



is equivalent to defining the symbol “*case*” as an abbreviation for “(*nom* \vee *acc* \vee *gen* \vee *dat*).” More generally, sorts can be used as abbreviations for disjunctions of sorts (each of which in turn may be either a species name or an abbreviation). Thus, for example, the sort *head* may be used to abbreviate the disjunction of the parts of speech *noun*, *verb*, *adj*, *prep*, *det*, *conj*, *deg* (for the sake of this example, we assume that these are all the parts of speech). But among the parts of speech we might want to group the first four together as *substantive* and the last three as *functional*. (Typically we group species or sorts together when there is some feature they have in common, or there is some constraint that applies to just them.) We can do this by positing the sort hierarchy



Given two sorts σ_1 and σ_2 we say σ_1 is a subsort of σ_2 if it is more specific than σ_2 in the sort hierarchy. Thus, e.g., *prep* and *functional* are subsorts of *head*. Alternatively, if σ_1 is a subsort of σ_2 , we can also say σ_2 is a supersort of σ_1 .

Note: for any sort hierarchy, it is always assumed that the subsorts are pairwise incompatible (mutually exclusive). Thus, positing a hierarchy amounts logically to adding to the grammar a set of constraints of the form:

$$((\sigma \leftrightarrow (\sigma_1 \vee \dots \vee \sigma_n)))$$

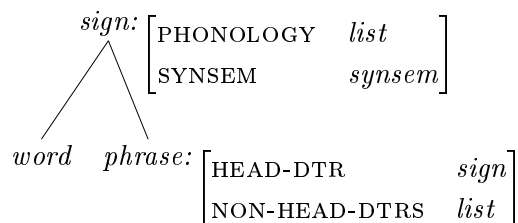
and

$$\neg(\sigma_i \wedge \sigma_j) \text{ (for all } i, j = 1, \dots, n, \text{ with } i \neq j)$$

2.19 Feature declarations

Feature declarations, employed in conjunction with sort hierarchies, are a convenient notational alternative to feature geometry constraints; feature declarations together with sort hierarchies are the usual way to express ontological constraints informally.

For example, the constraint ($\textit{noun} \rightarrow (\textit{CASE} : (\textit{nom} \vee \textit{acc} \vee \textit{gen} \vee \textit{dat}))$) is expressed as the feature declaration *noun*: [CASE *case*], where the sort hierarchy headed by *case* is defined as above. To take a more complex example, the sort hierarchy with feature declarations:



is equivalent to the following set of feature geometry constraints (assuming that *word*, *phrase*, and *synsem*, and *list* are species):

$$((\textit{word} \vee \textit{phrase}) \leftrightarrow ((\textit{PHONOLOGY} \doteq \textit{PHONOLOGY}) \wedge (\textit{SYNSEM} \doteq \textit{SYNSEM})))$$

$$(\textit{phrase} \leftrightarrow (\textit{DTRS} \doteq \textit{DTRS}))$$

$$((word \vee phrase) \rightarrow ((PHONOLOGY: list) \wedge (SYNSEM: synsem)))$$

$$(phrase \rightarrow ((HEAD-DTR: word \vee phrase) \wedge (NON-HEAD-DTRS: list)))$$

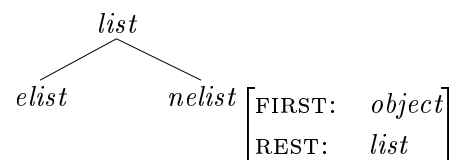
2.20 Lists

In HPSG, lists are usually introduced in the following way. We assume species names *elist* (empty list) and *nelist* (nonempty list), and feature names `FIRST` and `REST`, subject to the following feature geometry constraints:

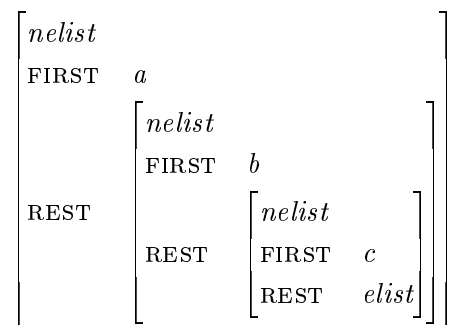
$$(nelist \leftrightarrow ((FIRST \doteq FIRST) \wedge (REST \doteq REST)))$$

$$(nelist \rightarrow (REST: (elist \vee nelist)))$$

Informally, we can say the same thing with a sort hierarchy and feature declarations, as follows:



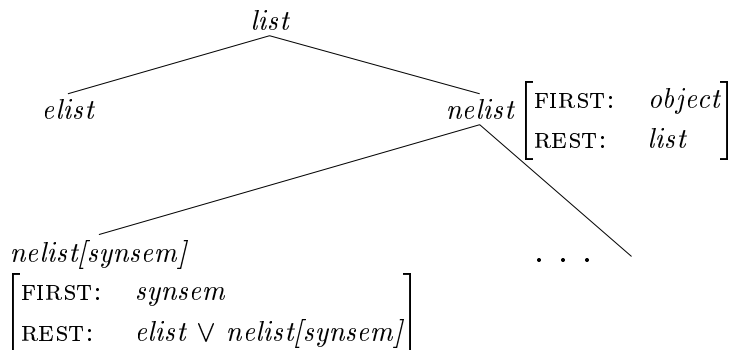
where *object* is a sort that abbreviates the disjunction of all the species names. (Thus if all the sorts are arranged in a single hierarchy, *object* is at the top.) The intuition is just that a list can be either an empty list or a nonempty list, and in the latter case it has a `FIRST`, which can be anything, and a `REST`, which is a list.⁵ In AVMS, list descriptions are usually abbreviated by angle-bracket notation. Thus, $\langle a, b, c \rangle$ abbreviates the AVMS:



Also, in AVMS, *elist* is often written $\langle \rangle$.

In practice, we usually want to work with lists all of whose members are of the same sort. In order to do this, we would need to revise the sort hierarchy to contain feature declarations along the following lines, with a new species of nonempty list $nelist[\sigma]$ for each species (or sort) of thing σ that we need lists of, as follows:

⁵The LISP notions `LIST`, `NIL`, `CONS`, `CAR`, and `CDR` correspond to *list*, *elist*, *nelist*, `FIRST` and `REST`, respectively.



2.21 A new kind of formula: path relations

For each natural number n , we introduce a finite set of *n-ary relation names*. If R is an n -ary relation name and π_1, \dots, π_n are paths, then $R(\pi_1, \dots, \pi_n)$ is a formula. Of course, to be useful, we must be told what it means for a feature structure to satisfy a path relation formula. There are a number of different proposals about how to do this (e.g. Richter in preparation), not all of which have the same consequences.

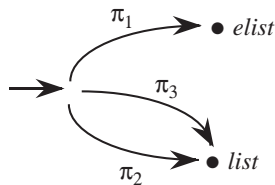
Append

Append is a ternary relation symbol. Path relation formulas employing *Append* are much-used in HPSG since they are used to express that one list is the concatenation of two other lists. More precisely, satisfaction for *Append* formulas is defined in such a way that, for any three paths π_1, π_2, π_3 ,⁶

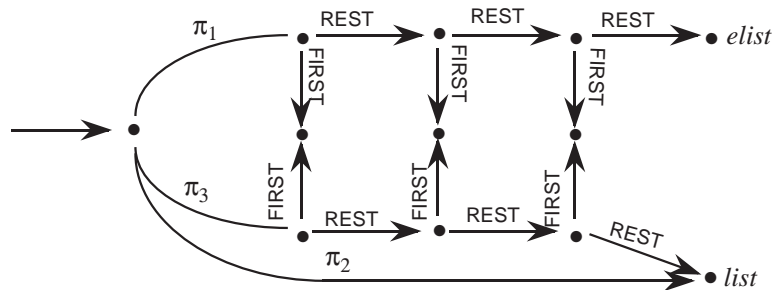
$\mathcal{G} \models \text{Append}(\pi_1, \pi_2, \pi_3)$ iff

1. $\mathcal{G} \models ((\pi_1 : \text{list}) \wedge (\pi_2 : \text{list}) \wedge (\pi_3 : \text{list}))$ and
2. either
 - a. $\mathcal{G} \models ((\pi_1 : \text{elist}) \wedge (\pi_2 \doteq \pi_3))$ or
 - b. $\mathcal{G} \models (\pi_1|_{\text{FIRST}} \doteq \pi_3|_{\text{FIRST}})$ and $\mathcal{G} \models \text{Append}(\pi_1|_{\text{REST}}, \pi_2, \pi_3|_{\text{REST}})$

The two options are illustrated by the following two schematic graphs:



⁶The following biconditional statement does not actually define what it means for a feature structure to satisfy an *Append*-formula, but it places a very strong constraint on possible definitions.



Append-SYNSEMS

The ternary relation symbol *Append-SYNSEMS* is employed in an important constraint called the Valence Principle. Intuitively speaking, satisfaction for *Append-SYNSEMS* formulas is defined in such a way that $\mathcal{G} \models \text{Append-SYNSEMS}(\pi_1, \pi_2, \pi_3)$ if π_1, π_2 and π_3 are all lists and π_3 is the concatenation of the list of SYNSEM values of the π_1 list with the π_2 list. More precisely, it is defined in such a way that the following biconditional holds:

$\mathcal{G} \models \text{Append-SYNSEMS}(\pi_1, \pi_2, \pi_3)$ iff

1. $\mathcal{G} \models ((\pi_1 : list) \wedge (\pi_2 : list) \wedge (\pi_3 : list))$ and
2. either
 - a. $\mathcal{G} \models ((\pi_1 : elist) \wedge (\pi_2 \doteq \pi_3))$ or
 - b. $\mathcal{G} \models ((\pi_1 : nelist) \wedge (\pi_1|FIRST|SYNSEM = \pi_3|FIRST) \wedge \text{Append-SYNSEMS}(\pi_1|REST, \pi_2, \pi_3|REST))$

Schematically:

$$\left[\begin{array}{l} \pi_1 < [\text{SYNSEM } \boxed{1}], [\text{SYNSEM } \boxed{2}], [\text{SYNSEM } \boxed{3}] > \\ \pi_2 \quad \boxed{4} \\ \pi_3 < \boxed{1}, \boxed{2}, \boxed{3}\boxed{4} > \end{array} \right]$$

The Valence Principle is the following constraint:

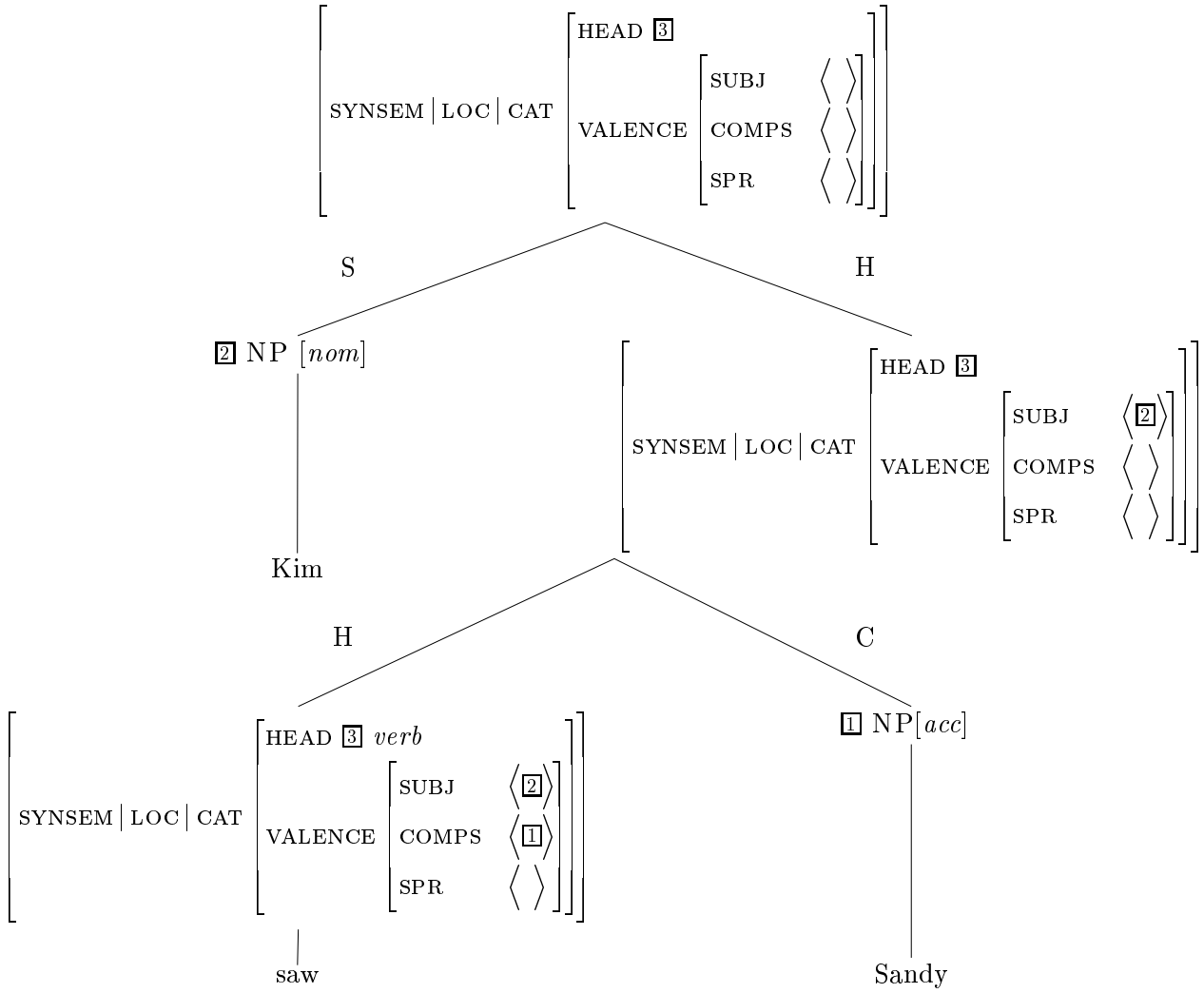
$$\text{headed-phrase} \rightarrow \left[\begin{array}{l} \text{SYNSEM|LOC|CAT|VAL} \left[\begin{array}{l} \text{SUBJ} \quad \boxed{4} \\ \text{COMPS} \quad \boxed{5} \\ \text{SPR} \quad \boxed{6} \end{array} \right] \\ \\ \text{HEAD-DTR|SYNSEM|LOC|CAT|VAL} \left[\begin{array}{l} \text{SUBJ} \quad \boxed{7} \\ \text{COMPS} \quad \boxed{8} \\ \text{SPR} \quad \boxed{9} \end{array} \right] \\ \\ \text{SUBJ-DTR} \quad \boxed{1} \\ \text{COMP-DTRS} \quad \boxed{2} \\ \text{SPR-DTR} \quad \boxed{3} \\ \text{Append-SYNSEMS}(\boxed{1}, \boxed{4}, \boxed{7}) \\ \text{Append-SYNSEMS}(\boxed{2}, \boxed{5}, \boxed{8}) \\ \text{Append-SC SYNSEMS}(\boxed{3}, \boxed{6}, \boxed{9}) \end{array} \right]$$

This says that, in a headed phrase, for each valence feature F, phrase, the F value of the head daughter is the concatenation of the list of SYNSEM values of the F-daughters with the F-value of the phrase itself.

An alternative “functional” notation for AVMs employing relations is illustrated in the following expression of the Valence Principle”

$$\left[\begin{array}{l} \text{SYNSEM|LOC|CAT|VAL} \left[\begin{array}{l} \text{SUBJ} \quad \boxed{4} \\ \text{COMPS} \quad \boxed{5} \\ \text{SPR} \quad \quad \boxed{6} \end{array} \right] \\ \text{HEAD-DTR|SYNSEM|LOC|CAT|VAL} \left[\begin{array}{l} \text{SUBJ} \quad \textit{Append-SYNSEMS}(\boxed{1}, \boxed{4}) \\ \text{COMPS} \quad \textit{Append-SYNSEMS}(\boxed{2}, \boxed{5}) \\ \text{SPR} \quad \quad \textit{Append-SYNSEMS}(\boxed{3}, \boxed{6}) \end{array} \right] \\ \text{SUBJ-DTR} \quad \boxed{1} \\ \text{COMP-DTRS} \quad \boxed{2} \\ \text{SPR-DTR} \quad \boxed{3} \end{array} \right]$$

This following (conventionalized) description of a sentence illustrates the effect of the Valence Principle:



REFERENCES

- Brody, Michael. 1995. Lexico-logical form. Linguistic Inquiry Monograph. Cambridge, MA.: MIT Press.
- Jackendoff, Ray S. 1996. The architecture of the language faculty. Cambridge, MA: MIT Press.
- King, Paul. 1989. A logical formalism for head-driven phrase-structure grammar. Manchester University Ph.D. Dissertation.
- Koster, Jan. 1987 Domains and dynasties. Dordrecht: Foris.
- Ladusaw, William. 1988. A proposed distinction between level and stratum. Linguistics in the morning calm 2, ed. by the Linguistic Society of Korea. Seoul: Hanshin Publishing Co.
- Moshier, Drew. 1988. Extensions to unification grammars for the description of programming languages. University of Michigan Ph.D. dissertation.
- Pullum, Geoffrey K. 1989. Formal linguistics meets the boojum. Natural Language and Linguistic Theory 7: 137-143.
- Richter, Frank. In preparation. Eine formale Sprache für HPSG und ihre Anwendung in einem Syntaxfragment des Deutschen. Ph.D. thesis, Eberhard-Karls-Universität, Tübingen.