[Warning: This file contains only the abstract and Chapter 1.
The full thesis is also available.]

SMOOTHING A PROBABILISTIC LEXICON VIA SYNTACTIC TRANSFORMATIONS

Jason Michael Eisner

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

2001

Professor Mitch Marcus
Supervisor of Dissertation

Professor Val Tannen
Graduate Group Chair

Abstract

SMOOTHING A PROBABILISTIC LEXICON VIA SYNTACTIC TRANSFORMATIONS

Jason Michael Eisner

Supervisor: Professor Mitch Marcus

Probabilistic parsing requires a lexicon that specifies each word's syntactic preferences in terms of probabilities. To estimate these probabilities for words that were poorly observed during training, this thesis assumes the existence of arbitrarily powerful transformations (also known to linguists as lexical redundancy rules or metarules) that can add, delete, retype or reorder the argument and adjunct positions specified by a lexical entry. In a given language, some transformations apply frequently and others rarely. We describe how to estimate the rates of the transformations from a sample of lexical entries. More deeply, we learn which properties of a transformation increase or decrease its rate in the language. As a result, we can smooth the probabilities of lexical entries. Given enough direct evidence about a lexical entry's probability, our Bayesian approach trusts the evidence; but when less evidence or no evidence is available, it relies more on the transformations' rates to guess how often the entry will be derived from related entries.

Abstractly, the proposed "transformation models" are probability distributions that arise from graph random walks with a log-linear parameterization. A domain expert constructs the parameterized graph, and a vertex is likely according to whether random walks tend to halt at it. Transformation models are suited to any domain where "related" events (as defined by the graph) may have positively covarying probabilities. Such models admit

a natural prior that favors simple regular relationships over stipulative exceptions. The model parameters can be locally optimized by gradient-based methods or by Expectation-Maximization. Exact algorithms (matrix inversion) and approximate ones (relaxation) are provided, with optimizations. Variations on the idea are also discussed.

We compare the new technique empirically to previous techniques from the probabilistic parsing literature, using comparable features, and obtain a 20% perplexity reduction (similar to doubling the amount of training data). Some of this reduction is shown to stem from the transformation model's ability to match observed probabilities, and some from its ability to generalize. Model averaging yields a final 24% perplexity reduction.

Chapter 1

Overview: An Executive Summary

1.1 Context of the Work

Humans are experts in the languages that they speak. So too must be machines that use language. Yet computational linguists are increasingly loath to build language-specific "expert systems" by hand. Such systems tend to be both expensive and fragile.

Instead, the dominant strategy of the past decade has been to build machines that learn statistically from raw or annotated language data. The computational linguist's job is to design an appropriately expressive statistical model whose parameters can be estimated, using appropriate algorithms, from a reasonable amount of data. These algorithms then learn the nuances of a particular language by tuning the parameters of the model to match known data from the language.

To increase the linguistic expertise of such machines, one must design statistical models that are sensitive to additional linguistic phenomena. But parameter estimation is harder for more complex models—so in practice it is a challenge to add linguistic sophistication that helps more than it hurts.

In the domain of syntax, the community has been adding sophistication carefully, one step at a time. Statistical models have been extended to capture a succession of linguistic phenomena, usually well enough in practice to exploit them for performance gains:

1. collocations (n-gram Markov models)

2. word categories (hidden Markov models)

3. trees (probabilistic context-free grammars)

4. subcategorization (lexicalized context-free grammars)

5. selectional preferences (various bilexical grammars)

6. long-distance movement (statistical gap passing)

7. arbitrary tree properties (stochastic LTAG, DOP, log-linear LFG, etc.)

These are mainly models of what Chomsky calls the "surface structure" of syntax trees.

This thesis builds on that progress by offering a working statistical treatment of Chomsky's "deep structure"—the transformational relationships *among* syntactic constructions that are related in meaning. We will see that a statistical model can detect such relationships in an existing sample of syntactic constructions, and use them to better predict the probabilities of constructions in the sample as well as new constructions that it has never seen before. It is in this sense that the model does "transformational smoothing" of probabilities.

In short, the model finds and applies language-specific generalizations about deep syntax. The approach is declarative. First we quantify a notion of a good grammar. Then we search in the continuous space of all possible grammars, trying to maximize the Bayesian or MDL product

$$\Pr(\text{training data} \mid \text{grammar}) \cdot \Pr(\text{grammar}) \tag{1.1}$$

The first factor emphasizes description, the second generalization. The first factor is high if the grammar assigns high probability to the constructions observed so far. The second, trickier factor is the *a priori* probability of the grammar itself; we will define it to be high if the grammar has strong, consistent internal structure with few stipulative exceptions.

The rest of this chapter is an overview of the scope and content of the work. It concludes with a detailed guide to the rest of the thesis.

1.2 Motivations

Let us begin by motivating the work from the interests of four different fields: engineering, linguistics, statistics, and child language acquisition.

1.2.1 The Engineering Problem: Learning More Syntax from Less Data

1.2.1.1 Probabilistic Parsing

Accurate parsing of natural language—discovering the recursive structure of an arbitrary sentence—is a major roadblock for systems that attempt to understand or translate text. A great many word meanings and grammatical constructions are *possible* (if rare) in the language. The typical sentence therefore has combinatorially many parses. The task is to output the *right* parse.

The trick is to search not for *possible* parses but for *probable* ones. A dynamic-programming parser builds a parse tree by composing context-free rules, or other small "chunks" of tree, according to some theory of syntax. One can define the tree's probability in terms of the probabilities of its component chunks (§5.3.3). The dynamic program can then find the most probable tree that parses a given input. A parser with access to probabilities can also run faster, by heuristically ignoring low-probability regions of the search space.

Probably the best-known recent English parsers of this sort are (Collins, 1997) and (Charniak, 2000), which are interesting, accurate and robust. However, this framework boasts many other practical and theoretical papers worthy of mention, and the techniques have been ported beyond English, e.g., (Collins et al., 1999).

1.2.1.2 The Need to Generalize

The trouble is that the chunk probabilities are hard to estimate from even a large sample of (expensively annotated) data. Many chunks that are needed to parse test data correctly are never observed at all in training data. In our data, derived from *The Wall Street Journal* via the Penn Treebank (§6.2), a remarkable 49% of sentences in the development set required for their correct parse at least one subcategorization frame (broadly construed

to include slots for all arguments and adjuncts) that never appeared with *any* headword in training data (§6.3). The parser cannot recover the correct parse if it assigns these novel frames a zero or negligible probability.

Hence, the parser must be able to generalize from chunks it has seen during training to new chunks that it has never seen. The list of chunks that actually appears in training data tends to be large, unorganized, redundant, and—most harmfully—incomplete. To parse, we need to reconstruct the *true* list of chunks in the language, and their true probabilities, as well as possible. This means looking at internal chunk structure.

Parsers in the literature usually deal with this problem by building the "atomic" chunks (or equivalently the parse trees) from smaller "subatomic" units that are more likely to recur in training data: binary-branching rules or single dependencies rather than complete subcategorization frames. However, even weak versions of this atom-splitting lead to overly strong independence assumptions (§6.6.3). Other generative models for chunks have occasionally been considered (Collins, 1997, model 2), although these were not the best performers in our experiments either (§6.7.1).

This thesis explores a more sophisticated and apparently better-performing approach to generalizing from old (observed) chunks to new ones. The approach is inspired by both linguistic tradition (§1.2.2) and an analysis of the data (§2.4.2.1).

1.2.1.3 An Example

Concretely, consider the table in Table 1.1. It shows 6 words with all the subcategorization frames they headed in our training data.[1] (Since the words are uninflected verbs, they only appear in infinitive or present-tense form in our unmorphologized data.) The symbol "——" stands for the headword in the subcategorization frame. For example, the second row says the lexicalized context-free rule `S` → `To encourage NP PP` was used once in the hand-constructed training parses, while `S` → `To fund NP PP` was used twice, and so forth.

In order to parse, we need to replace the counts in this table with probabilities Pr(frame | headword), as shown in Table 1.3, such that

[1] For reasons discussed in §2.4.1, this thesis interprets "subcategorization frame" broadly to include all dependents—including adjuncts (such as some `PP`'s) and subjects, which have traditionally been excluded from subcategorization frames. To avoid confusion we will simply use the term "frame" in future.

	encourage	question	fund	merge	repay	remove
S → To —— NP	1	1	5	1	3	2
S → To —— NP PP	1	1	2	2	1	1
S → To AdvP —— NP						1
S → To AdvP —— NP PP						1
S → NP —— NP .		2				
S → NP —— NP PP .	1					
S → NP Md —— NP	1					
S → NP Md —— NP PP-TMP					1	
S → NP Md —— PP PP						1
S → To —— PP				1		
S → To —— S	1					
S → NP —— SBAR .		2				

Table 1.1: Counts of all the S → ··· frames that appear with six particular words in training data. The words were not specially chosen, except that they were all required to appear with S → To —— NP PP (so that they would be related) and to appear 4–7 times (so that this example table would be small but not trivial).

Pr(*RHS* | *headword*, *LHS*)	encourage	question	fund	merge	repay	remove
S → To —— NP	0.200	0.167	0.714	0.250	0.600	0.333
S → To —— NP PP	0.200	0.167	0.286	0.500	0.200	0.167
S → To AdvP —— NP	0	0	0	0	0	0.167
S → To AdvP —— NP PP	0	0	0	0	0	0.167
S → NP —— NP .	0	0.333	0	0	0	0
S → NP —— NP PP .	0.200	0	0	0	0	0
S → NP Md —— NP	0.200	0	0	0	0	0
S → NP Md —— NP PP-TMP	0	0	0	0	0.200	0
S → NP Md —— PP PP	0	0	0	0	0	0.167
S → To —— PP	0	0	0	0.250	0	0
S → To —— S	0.200	0	0	0	0	0
S → NP —— SBAR .	0	0.333	0	0	0	0
(other)	0	0	0	0	0	0

Table 1.2: The probabilistic lexicon most likely to generate the counts in Table 1.1. While it is trivial to compute, unfortunately it is an improbable lexicon *a priori*. It would be a poor engineering choice since it assigns 0 probability to many plausible frames.

Pr(*RHS* | *headword*, *LHS*)	encourage	question	fund	merge	repay	remove
S → To — NP	0.088	0.043	0.398	0.127	0.171	0.131
S → To — NP PP	0.030	0.016	0.101	0.114	0.032	0.048
S → To AdvP — NP	0.00017	0.000064	0.00019	0.00010	0.00020	0.033
S → To AdvP — NP PP	0.000057	0.000023	0.000049	0.000092	0.000038	0.012
S → NP — NP .	0.013	0.104	0.0021	0.0053	0.0063	0.0034
S → NP — NP PP .	0.016	0.0068	0.00095	0.0019	0.0021	0.0010
S → NP Md — NP	0.023	0.0020	0.0055	0.0030	0.035	0.034
S → NP Md — NP PP-TMP	0.00056	0.000061	0.000031	0.000079	0.0047	0.00030
S → NP Md — PP PP	0.00014	0.000032	0.000024	0.000089	0.00029	0.0018
S → To — PP	0.016	0.015	0.052	0.112	0.031	0.051
S → To — S	0.034	0.0061	0.010	0.016	0.012	0.0053
S → NP — SBAR .	0.018	0.111	0.0029	0.0076	0.0089	0.0048
(other)	0.762	0.696	0.428	0.613	0.695	0.675

Table 1.3: Part of the probabilistic lexicon induced from training data by the best model from previous literature, a smoothed bigram model (§6.6.1.2). There exist many more rows (in fact, infinitely many) and columns. Each column sums to 1 (if we consider only the S → ⋯ rows).

Pr(*RHS* | *headword*, *LHS*)	encourage	question	fund	merge	repay	remove
S → To — NP	0.142	0.117	0.397	0.210	0.329	0.222
S → To — NP PP	0.077	0.064	0.120	0.181	0.088	0.080
S → To AdvP — NP	0.00055	0.00047	0.0011	0.00082	0.00091	0.079
S → To AdvP — NP PP	0.00018	0.00015	0.00033	0.00037	0.00026	0.050
S → NP — NP .	0.022	0.161	0.0078	0.0075	0.0079	0.0075
S → NP — NP PP .	0.079	0.0085	0.0026	0.0027	0.0026	0.0026
S → NP Md — NP	0.090	0.0021	0.0024	0.0020	0.024	0.0026
S → NP Md — NP PP-TMP	0.0018	0.00016	0.00017	0.00016	0.069	0.00019
S → NP Md — PP PP	0.00010	0.000027	0.000027	0.000038	0.000078	0.059
S → To — PP	0.0092	0.0065	0.012	0.126	0.010	0.0091
S → To — S	0.098	0.0016	0.0043	0.0039	0.0036	0.0027
S → NP — SBAR .	0.0034	0.190	0.0032	0.0032	0.0032	0.0032
(other)	0.478	0.449	0.449	0.461	0.461	0.482

Table 1.4: Part of the probabilistic lexicon induced from training data by transformational smoothing.

1. Each column of the table sums to 1.

2. The observed counts constitute a likely sample from the probabilities.

3. The pattern of probabilities is *a priori* plausible.

The result is called a conditionalized "probabilistic lexicon."

The maximum-likelihood estimate (Table 1.2) satisfies the first two properties but not the third; it is too stipulative. A smoothing method such as our transformational smoothing (Table 1.4) is designed to "smear" these estimates down the column, generalizing from one row to another. This lets it place non-zero probabilities on rules that appear in the correct parses of test data, including novel rules.

We would like a smoothing method to place high probabilities on the test rules so that it can correctly choose the correct parses, which contain those rules. Thus, our cross-entropy evaluation measure is (the log of) the product of the test probabilities (§6.1).

1.2.2 The Linguistic Problem: Stochasticizing Deep Structure

1.2.2.1 Background: Lexical Redundancy

Now let us turn to the linguistic motivation. Many modern theories of syntax are *lexicalist*. They agree with the dynamic-programming parser of §1.2.1.1: a legal syntax tree is formed by composing "chunks" of syntax, each of which is governed by some lexical item. The collection of chunks is called the **lexicon**, and each individual chunk is called a **lexical entry**.

The form of the chunks and the compositional mechanisms vary from theory to theory (see Fig. 2.1 on p. 36 for examples), but the basic idea remains the same: there is no language-specific grammar outside the lexicon.

This thesis responds to a problem faced by all such theories, known as "lexical redundancy." Suppose English has a lexical entry for the word `fippened` that calls for a subject and a direct object: *The beast fippened its dinner.* It is then almost certain that the English lexicon also lists a passive entry for `fippened`: *The dinner was fippened by a beast.* One might also accept reasonable odds on a bet that English lists an intransitive entry: *The beast fippened (all day)*, or perhaps, *The dinner fippened (slowly).*

The basis for these bets is that many *other* words in English, such as `cooked`, exhibit the same pattern of lexical entries.[2] A simple lexical theory would take these common patterns to be coincidences, and would have to stipulate them over and over in order to describe the behavior of `fippened`, `cooked`, and other words.

1.2.2.2 Lexical Redundancy Rules

Linguists prefer not to enlarge the grammar with such repeated stipulation. Occam's Razor calls for modeling the patterns somehow—both to improve the lexicon's elegance and to explain why a learner who observes the new word `fippened` in a single linguistic context is immediately comfortable using it in the other contexts predicted above.

So lexicalist theories generally allow rules that *derive* lexical entries from one another. Again, the form of these rules may vary from theory to theory (Fig. 2.1). They are the spiritual descendants of transformations in transformational grammar, the main difference being that they cannot modify any chunk bigger than a lexical entry, and do their work before the chunks are assembled. There is still no language-specific grammar outside the lexicon—but there is now a language-specific grammar *inside* the lexicon.

Thus, a grammar consists of a lexicon plus a system of lexical redundancy rules. The lexicon is still the final arbiter of syntax, since it may include both positive and negative exceptions to the rules. However, the rules encode the regular patterns that underlie most of the lexicon.

1.2.2.3 Introducing Rule Probabilities

What this thesis contributes to linguistics is a sensible *stochastic* treatment of these lexical redundancy rules, their exceptions, and their acquisition. Just as one can attach probabilities to the lexical entries—so as to better model competence and performance—we will attach probabilities to the rules.

[2] Transitive verbs' ability to passivize is almost without exception in English, although they do not all passivize equally often. The tendency of some transitive verbs to boast intransitive entries as well is weaker, but still systematic and worth learning. Such "Object Drop" entries are used in the Brown corpus (Kucera and Francis, 1967) about 55% of the time with `eat` and `drink`, 25% of the time with `finish`, and never with `devour`. It is useful for a learner to recognize that Object Drop is at least plausible for a novel verb, and then to be able to tune its rate of application from verb to verb.

The stochastic framework will provide an attractive perspective on a number of linguistic issues. In particular it allows a unified and gradient treatment of language-specific syntactic generalizations at all levels: exceptions, subregularities (patterns of exceptions), rules, and tendencies shared by many rules.

Just as important, the stochastic framework yields up statistical methods to *learn* all these generalizations and exceptions from data. Learning the rules is of particular interest, because they encode the deep structure of the language and because they often have semantic consequences.

1.2.2.4 An Example

Consider again the data of Table 1.1. There are clear structural relationships among the frames that appear with these six words.

The first two rows are heavily correlated, as discussed further in §1.2.3.3 below. They correspond to frames that are related by a simple edit: the addition of a `PP` (prepositional phrase) at the right edge. We can regard this as a `PP`-adjunction that transforms *to encourage loyalty* ⇒ *to encourage loyalty [with frequent-flier miles]*. More precisely, it transforms the lexical entry that licenses the former phrase into an entry that licenses the latter. Indeed, the data suggest that a word must have the first entry to have the second.[3]

The same transformation relates rows 3 and 4. It so happens that `remove` likes to be modified by an `AdvP` (adverbial phrase): *to* {*completely,properly,surgically,...*} *remove the asbestos*. The same `PP`-adjunction transformation that can be learned from rows 1 and 2 (and other data) predicts that `remove`'s mildly idiosyncratic entry in row 3 predicts its mildly idiosyncratic entry in row 4.

Meanwhile, rows 1 and 2 are themselves related to rows 3 and 4, by an `AdvP`-insertion transformation. Notice that this transformation tends to apply in certain contexts: it prefers to insert the `AdvP` immediately before the headword ——, even at the cost of splitting an infinitive, as here. We will see below how to model this preference.

Rows 1 and 2 are also related to rows 5 and 6, by a more linguistically interesting

[3]Having the second entry was required for a word to be included in Table 1.1; see the caption. The motivation for lexical redundancy rules is to explain the "coincidence" that these words all also appeared with the first entry as well.

sequence of transformations: *to encourage loyalty* $\Rightarrow^*$ *[Frequent-flier miles] encourage loyalty* . First `encourage` acquires an `NP` (noun phrase) subject at the left edge. A sentence with a subject is allowed to acquire tense as well, so a second transformation drops the infinitive marker `To`. And tensed sentences are allowed to serve as main sentences, so a third transformation can now add a final period.[4]

The next few rows undergo a similar process, except that they acquire tense differently: not by dropping the infinitive marker `To`, but by substituting a modal `Md` (*will*, *can*, *should* ...). This is a productive transformation in English.

Finally, the observations in the last three rows are lexically idiosyncratic exceptions. Unlike in previous rows, most of the zero counts here are not accidental gaps in the data, but actually reflect very low probabilities:

*to merge/*encourage/*question [with First Hospital]*

*to encourage/*question/*merge [students to consider teaching]*

*Experts question/*encourage/*merge [whether the magazine can risk it]* .

Though Table 1.1 shows only a small and somewhat artificial subset of the real data, the similarity of the frames is striking. Regarded as strings, they are all closely related by edit distance. If we ignore the edit needed to turn `PP` into `PP-TMP` (a temporal `PP`), then each frame is at edit distance 1 from some other frame, and only two of the frames achieve an edit distance of 3 from the first two frames. §2.4.2 will use this observation to define a particularly simple set of "edit" transformations to be used in the experiments.

Even the lexically idiosyncratic frames in the last few rows appear to be licensed by simple edit transformations. For example, the last row is a transformation of the fifth row: some verbs, like `question`, can type-shift their `NP` complement (*management's credibility*) to an `SBAR` complement (*whether the magazine can risk it*). The table shows that `question` tends to allow main-verb lexical enties of both sorts (but does not particularly like `PP` modifiers). So it is an idiosyncratic verb, but its idiosyncrasies do follow a pattern. We would like our statistical account of lexical redundancy rules to be able to capture such

[4] This particular sequence of three transformations is not the only reasonable way to describe this process. It would in any case be preferable to use separate LHS nonterminals $\texttt{S}_{\texttt{inf}}$ and $\texttt{S}_{\texttt{tensed}}$, so that period-insertion does not have to look non-locally in the frame's RHS to discover whether it is licensed. (Separate LHS nonterminals would also be desirable because other context-free rules subcategorize for one or the other.) See §5.5.4.3 for more on how transformations that manipulate features might work.

semi-productive processes.

1.2.3 The Statistics Problem: Capturing Covariance of Related Events

In language and in life,[5] one often needs to estimate a probability distribution over a finite or countably infinite set of mutually exclusive events. Some of the events in this set might be related in ways that are known *a priori* to a domain expert. For example, linguistic constructions may be related by lexical redundancy rules, web pages may be related by links, movies may be related by their common personnel, and so forth.

One therefore wants a model that can, if necessary, capture the fact that related events have related probabilities. Every type of *potential* relationship in the domain should correspond to a parameter that models the *actual* strength of such relationships in the data.

1.2.3.1 Previous Approaches

An obvious attempt is to use a log-linear model, also known as a maximum-entropy or Gibbs distribution. Each event is characterized by a set of features. Increasing a feature's weight will equally scale up the probabilities of all events bearing that feature (after which the probability distribution over the full set of events must be renormalized).

However, this is not quite a solution. The log-linear model learns probabilities, not relationships. A feature weight does not really model the strength of the relationship between two events e, e'. It only influences both events' probabilities. If the probability of e is altered by some *unrelated* factor (another feature), then the probability of e' does not respond unless it happens to share this additional factor (feature). So the relationship between e and e' is a 100% correlation if all other parameters are fixed,[6] but weakens quickly as the other parameters are allowed to vary.

An alternative is to use a directed graphical model (Bayesian network). Such a model can directly capture the causal relationship between e and e'. However, a graphical model has the wrong form. It describes a joint distribution over multiple random variables—so

[5] My life, anyway, and probably the reader's.

[6] Assuming that the feature with variable weight appears just as often on e' as on e. More generally, if it appears $0, 1, 2, \ldots$ times as often on e', then the relation will be absent, linear, quadratic, etc. The number of appearances of each feature on each event is traditionally set by a domain expert who defines the features, although learning it is an interesting possibility.

e and e' are not mutually exclusive and their probabilities may sum to more than 1—whereas we are interested in the distribution over mutually exclusive values $e, e', \ldots$ of a single random variable. In addition, graphical models disallow causal cycles. So they cannot model a situation where extraneous factors affecting e' will affect e *as well as* vice-versa;[7] furthermore, learning the direction of causality requires learning the discrete topology of the model.

1.2.3.2 Transformation Models

This thesis tries to fill the bill by introducing a class of "transformation models," together with natural priors and parameter estimation algorithms. The conceit in transformation models is that events can spontaneously transform into one another. If e regularly transforms into e', then any effects that raise $\Pr(e)$ also raise $\Pr(e')$. The strength of the relationship between events e and e' is represented by how often e transforms into e'.

The distribution being modeled is presumed to be a snapshot of the events after all transformations have stopped. Observing this distribution provides evidence from which we infer the parameters of the underlying transformational process. In particular, suppose that *a priori*, $e_1, e_2, e_3, \ldots$ have some relationship to $e'_1, e'_2, e'_3, \ldots$ respectively, and that empirically, variance among the $\Pr(e_i)$ explains part of the variance among the $\Pr(e'_i)$. Then we have evidence that this type of relationship is strong and each e_i tends to transform into e'_i.

Transformation models do have resemblances to log-linear and graphical models. Each relationship's strength (i.e., each transformation's probability) is modeled log-linearly in terms of features of the relationship; in fact log-linear models are a special case of transformation models.[8] And since $e' \Rightarrow e$ may have a different probability than $e \Rightarrow e'$, the model describes asymmetric causal relationships just as directed graphical models do, with

[7] Unless one adds arcs to e from all ancestors of e', and so forth. But then it would require a complex parameterization to ensure that these arcs were mediated by the strength of the relationship between e' and e.

[8] To convert a log-linear model into a transformation model, allow the initial event START to transform randomly into any other event (where the process then stops), with a probability that depends log-linearly on features of the latter. The probability of this transformation is then the probability of the event. See §7.3.4.2 for a kind of converse: a variant kind of transformation model can be regarded as a log-linear model over events that have been augmented with derivational history.

similar conditional independence and abductive ("explaining away") properties.

A terminological note: Discussions of this work can help support web searches by sticking to the term "transformation model" rather than "transformational model," as the latter collocation is already in online use (e.g., in nursing). I do however use "transformational smoothing" to refer to the effect of a transformation model.

1.2.3.3 An Example

There is a relationship between rows 1 and 2 of Table 1.1. Across the board, row 2 is (let us say) about half of row 1. We can model such a relationship by saying that for every three instances of $\texttt{S} \rightarrow \texttt{To} \text{ —— } \texttt{NP}$, one transforms into $\texttt{S} \rightarrow \texttt{To} \text{ —— } \texttt{NP PP}$, leaving the observations in a 2:1 ratio.

Of course, row 2 is not *exactly* half of row 1. Three explanations are available within the model:

- The counts are merely samples of the true probabilities. The small counts observed in rows 1–2 are certainly consistent with a true 2:1 ratio. This explanation becomes less attractive as the counts get large.

- Row 2 is not solely derived from row 1. There may exist other transformations that can also produce row 2's $\texttt{S} \rightarrow \texttt{To} \text{ —— } \texttt{NP PP}$. If the probabilities of all such transformations are held fixed while others vary, then row 2's $\Pr(\texttt{S} \rightarrow \texttt{To} \text{ —— } \texttt{NP PP})$ is a fixed linear combination of row 1's $\Pr(\texttt{S} \rightarrow \texttt{To} \text{ —— } \texttt{NP})$ *and* the probabilities of other entries. While $\frac{1}{2}$ may be the exact coefficient of row 1's $\Pr(\texttt{S} \rightarrow \texttt{To} \text{ —— } \texttt{NP})$ in this linear combination, other summands exist and affect row 2's $\Pr(\texttt{S} \rightarrow \texttt{To} \text{ —— } \texttt{NP PP})$ as well. So the *ratio* of the two probabilities is not $\frac{1}{2}$, although they are 100% correlated. (The slope of the regression line is $\frac{1}{2}$ but the intercept is not 0.)

- The six transformations that turn the six row-1 entries into their corresponding row-2 entries have probabilities that are loosely tied (because they share many features) but need not be identical. $\texttt{S} \rightarrow \texttt{To merge NP}$ may transform into $\texttt{S} \rightarrow \texttt{To merge NP PP}$ with some probability $> \frac{1}{3}$, thanks to headword-specific features that model the

exceptional properties of `merge`, in particular its tendency to take a `PP` argument (*with* `NP` or *into* `NP`).

1.2.4 The Language-Learning Problem

The long-term motivation of this thesis is to contribute to models of language learning by human or computer. This section speculates on how this work might fit into that program.

The Achilles' heel of current statistical parsing work is its dependence on a corpus of sample parses. One would like to reduce or eliminate that dependence. Children learn language from raw speech signals, situated in a real-world environment. Could a computer induce a grammar from raw text? What statistical biases would help it do so?

1.2.4.1 Enriching EM with a Transformational Prior

For learning context-free grammars from raw text, Inside-Outside reestimation (Baker, 1979) looms large in the intellectual landscape. Yet it works poorly on language text (Lari and Young, 1990; de Marcken, 1995).

One kind of improvement is to use a prior distribution over possible grammars. Since Inside-Outside is an instance of the EM algorithm, it is possible to constrain it with a prior. Using a linguistically sensible prior may help align EM's goal of maximizing probability with language learning's goal of maximizing interpretability.

A transformation model of the lexicalized grammar offers a natural prior, which says that low-cost lexicons—in §§1.3.2–1.3.3's sense of having broad generalizations with few exceptions—are *a priori* more likely. One could also modify the prior to include substantive linguistic biases.

1.2.4.2 Failure-Driven Learning

A prior by itself is unlikely to rescue EM. One also needs some way of avoiding local maxima. Let us consider a variant of EM that grows a labeling of the data slowly from a small seed model. The strategy is inspired both by a reasonable perspective on human learning and by several successful unsupervised learning algorithms for NLP (see §8.2.4).

All the time that children are learning language from the examples they hear, they have a more pressing concern: trying to make sense of those examples. It is well known that children comprehend a wider variety of constructions than they can produce, or produce correctly. Indeed, the same is true of adults, who can successfully guess the syntax and semantics of novel words in context.

This suggests failure-driven learning: Suppose a child is unable to parse a sentence according to her current grammar. If the sentence is hopeless gibberish to her, she ignores it. But if the sentence lies at the edge of her grammar, so that a slight change to the grammar would let her parse it and hence understand it, then she has an incentive to make that change.

For example, *The beast fippened all day* requires a lexical entry e that licenses `fippened` as an intransitive verb (§1.2.2.1). If the child does not have such an entry, she may be still be able to settle on the best parse with reasonable confidence. This requires "stretching" the grammar and adding the new entry to it.

1.2.4.3 Transformational Plausibility in Failure-Driven Learning

Of course, there may be multiple ways to stretch the grammar. An alternative is to treat *all day* as a noun phrase rather than an adverbial. So for the child to have reasonable confidence in the first parse, she must consider it to require a much less painful stretch.

Hence, the child must estimate how common it is for `fippened` to be used intransitively: $\Pr(e \mid \texttt{fippened})$. If she has seen `fippened` a million times and never as an intransitive, then her estimate should be small (probably $< 10^{-6}$). But if she has seen `fippened` only once or twice, as a transitive verb, then she should be willing to back off to *other* verbs, using the fact that many transitive verbs in English can also be used intransitively.

The transformation model of this thesis can make exactly such a smoothed estimate. It interpolates in Bayesian fashion between the actual observations of `fippened` and the generalizations it has extracted about English as a whole.

Our perspective is that all possible lexical entries are always in the lexicon. The requisite entry e has been dormant in the lexicon all along, with some low probability. It may be recruited into action to parse the newly observed datum *The beast fippened all day.*

If the parse using e is much more probable overall than alternative parses, then an EM-style algorithm will consider it likely that e has now been observed. This new observation raises the probability of e for future use.

Similarly, Pinker (1989) notes that adults readily interpret novel lexical entries as the result of applying semi-productive transformations to known entries. Again, our perspective is that these entries were dormant in the lexicon all along.

1.3 The New Idea

1.3.1 A Transformation Model of the Lexicon

Triangulating from the various motivations in §1.2, the shape of the solution should now be clear.

If linguistic theory is right, then the distribution of chunks (lexical entries) in a parser's training data (§1.2.1) should provide statistical evidence of lexical redundancy transformations (§1.2.2). These transformations spontaneously select chunks for production and convert them into one another at various rates. Several can apply in sequence. Of course, these choices are not "really" spontaneous—they reflect the speaker's intentions—but they appear spontaneous to a listener or a model with no prior knowledge of those intentions.[9]

One can build a transformation model of this process (§1.2.3). The details of the model—the form of the lexical entries, the set of possible transformations, and the parameterization of transformation rates—should be chosen on linguistic grounds. They specify the substance of universal grammar. From a learning viewpoint (§1.2.4), they describe the kinds of generalizations that a learner should be sensitive to in training data.

Learning a particular language is then a matter of estimating the parameters of the model: the language-specific rates of the transformations. Once discovered, these non-zero rates predict the existence and probability of an unbounded number of new chunks. They also yield smoothed probability estimates for old chunks. A statistical parser (or parse reranker) can therefore request a probability estimate for any potential chunk it is considering.

[9] It is very common and very useful to account for the effects of unmodeled factors as some kind of random noise, even in a deterministic physical process (Kalman, 1960).

1.3.2 Stochastic Rules and Exceptions

We will have to attach probabilities to lexical redundancy rules. A typical low-probability rule in English is Yiddish-movement: *Breakfast, the beast fippened rarely.* It is a completely regular process but simply does not apply very often, so that the entry it produces for `fippened` (or any verb) has low probability, is rarely used in generation, and is avoided in comprehension (when possible) in favor of better entries. A high-probability rule corresponds to an obligatory or nearly-obligatory transformation.

The plan that Sofia will swallow illustrates the use of such probabilities in comprehension. To disambiguate this `NP`, the hearer must ask herself, among other things, whether *swallow* is more likely to extract its object or drop it.[10] Evidence for the extraction reading is that the extraction rule has higher probability in English—at least in the Penn Treebank, where more verbs undergo extraction than object drop, both by type and by token.

We treat not only rules but exceptions in a graded fashion. Taken together, the rules predict probabilities for all lexical entries (from the probabilities of other entries). An entry is exceptional *to the degree* that its probability deviates from prediction. Entries that are "listed" in the lexicon are simply entries that occur much more often than predicted. All other entries are derived.

We are concerned with the *cost* of a grammar rather than its size. The lexicon holds *every* possible lexical entry,[11] in the sense of assigning it a non-zero (but usually negligible!) probability. This is perfectly fine so long as only a few of these probabilities need to be strongly listed and the rest are highly predictable.

Our cost model for a grammar follows Occam's Razor: it penalizes both generalizations and exceptions. Exceptions are expensive, so the lexicon avoids adding strong exceptions unless justified by a good deal of evidence. Adding a rule is expensive, for a learner or a linguist, but worth it if it makes several lexical entries less exceptional. In short, a linguist or learner will try to account for the data by deriving all lexical entries from a few listed entries with a few rules. The minimum-cost lexicon is the one that multiplies entities to

[10] On the object extraction reading, Sofia will swallow the plan; on the object drop reading, the plan's content is that Sofia will make a swallowing noise, perhaps to distract the guards.

[11] "Possible" according to the syntactic theory being used—that is, the Universal Grammar that determines the allowable form of entries and rules.

adjoin PP to S following NP	adjoin AdvP to S following NP	···
adjoin PP to S following $\bar{\text{S}}$	adjoin AdvP to S following $\bar{\text{S}}$	···
adjoin PP to S at left edge	adjoin AdvP to S at left edge	···
⋮	⋮	

Figure 1.1: A large set of rules characterized by a smaller set of underlying features (in this case, *what* is being adjoined and *where*).

necessity but not beyond.

1.3.3 Rules and Their Features

A twist is that we do not take the transformational rules to be the deepest elements of the grammar. Again, let us concern ourselves with the cost of the grammar rather than its size. The grammar includes *every* possible rule with some non-zero (but usually negligible!) probability. Again, this is fine so longer as the rules' probabilities are predictable from a small, finite set of rule features.

The motivation for rule features is to allow a more sensitive treatment of rule probabilities. Instead of having one rule each for PP-adjunction and AdvP-adjunction, we can have many of each kind, roughly as shown in Fig. 1.1. This improves both the descriptive and the explanatory power of the theory. By splitting PP-adjunction into several context-sensitive rules, which share only some of their features, we can assign them different probabilities according to *where* the PP is adjoined. And since rows as well as columns in Fig. 1.1 share features, we can also capture generalizations that cut across PP and AdvP: in English the probabilities decrease downward in the table, just as they decrease rightward. (In practice, we use tables with many more than two dimensions.)

Now it is not really adding a rule that is expensive; it is adding a feature. A feature is influential in the grammar to the extent that its weight is far from zero. So a linguist or learner will prefer to keep the number of influential features small. In other words, a low-cost grammar is one that is described by a few broad generalizations.

Finally, rule features can unify the treatment of rules and exceptions. We can take such a fine-grained view that PP-adjunction to two separate entries e and e' is regarded as

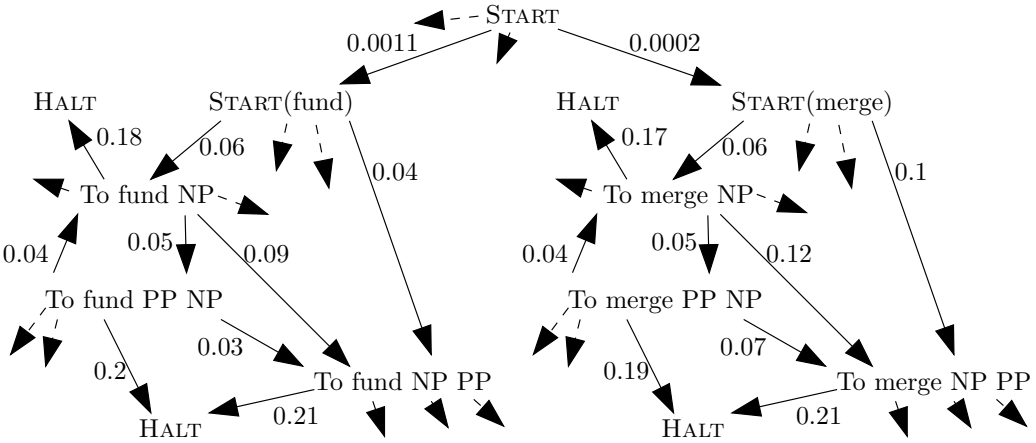

Figure 1.2: A fragment of a transformation model, showing arc probabilities. (Dashed arrows stand for other arcs not shown in this figure.)

two separate rules. Even if e and e' are identical except for their headword, the two rules' probabilities can differ if we have features that are specific to e and e'. These features suffice to describe exceptions.

So the grammar of English, say, is simply a collection of feature weights, which together describe exceptions, rules, and generalizations about the rules. Occam's Razor reduces to a simple principle: *keep all feature weights close to zero.* To learn a language, we try to find small feature weights that do a good job of modeling the data.

1.4 A Sketch of Transformation Models

We now see by example how the idea of the above section is turned into mathematics. For example, "low cost" in the above section will be realized as "high prior probability."

1.4.1 The Transformation Graph

Part of a transformation model for the lexicon is shown in Fig. 1.2. The vertices are linguistically possible lexical entries; the arcs (directed edges) represent linguistically possible transformations. The set of arcs leaving any given vertex has total probability 1.

A transformation model gives a language-specific probability distribution over the lexical entries. To sample from this distribution, take a random walk from the special vertex START to the special node HALT. The last lexical entry reached before HALT is the sample.

For example, the random walk might reach `S` → `To` `fund` `NP` in two steps and simply halt there. This happens with probability $0.0011 \cdot 0.06 \cdot 0.18$. Or, having arrived at `S` → `To` `fund` `NP`, it might transform it into `S` → `To` `fund` `PP` `NP` and then further to `S` → `To` `fund` `NP` `PP` before halting at the latter.

The probability distribution over lexical entries is entirely determined by the probabilities on the arcs. Crucially, if one increases the probability of an arc that goes to `S` → `To` `fund` `NP`, then the distribution allocates more probability mass not only to `S` → `To` `fund` `NP` but also to its child `S` → `To` `fund` `NP` `PP` (not to mention *its* children).[12] It is in this sense that the graph captures covariance among the entries' probabilities.

1.4.2 Parameter Tying

The graph in Fig. 1.2 is language-independent, but the arc probabilities are language specific. In principle the graph is infinite, since it assumes that lexical entries may take arbitrarily many descendants.

Does this mean that a language learner must estimate infinitely many language-specific probabilities, one for each arc in the graph? No, because many of the arcs have something in common. Arcs that represent the same transformation will be constrained to have the same probability. Arcs that represent similar transformations will be constrained to have similar probabilities.

To accomplish this, we parameterize the arc probabilities with a small, finite set of feature weights $\theta_1, \theta_2, \ldots \in \mathbb{R}$. The probabilities are defined in terms of these weights as shown in Fig. 1.3. Notice that this is a conditional log-linear (maximum-entropy) model of Pr(arc | vertex), i.e., of Pr(transformation | entry to be transformed).

For example, four transformations shown in Fig. 1.3 have the feature that they insert a prepositional phrase (`PP`) somewhere or other. That feature's weight θ_3 is used for all such transformations' arcs. The transformations available at each lexical entry compete with

[12] I.e., raising the probability of the arc to `S` → `To` `fund` `NP` increases the chances of sampling that entry and its descendants. The random walk is more likely to reach them, hence more likely to halt at them.

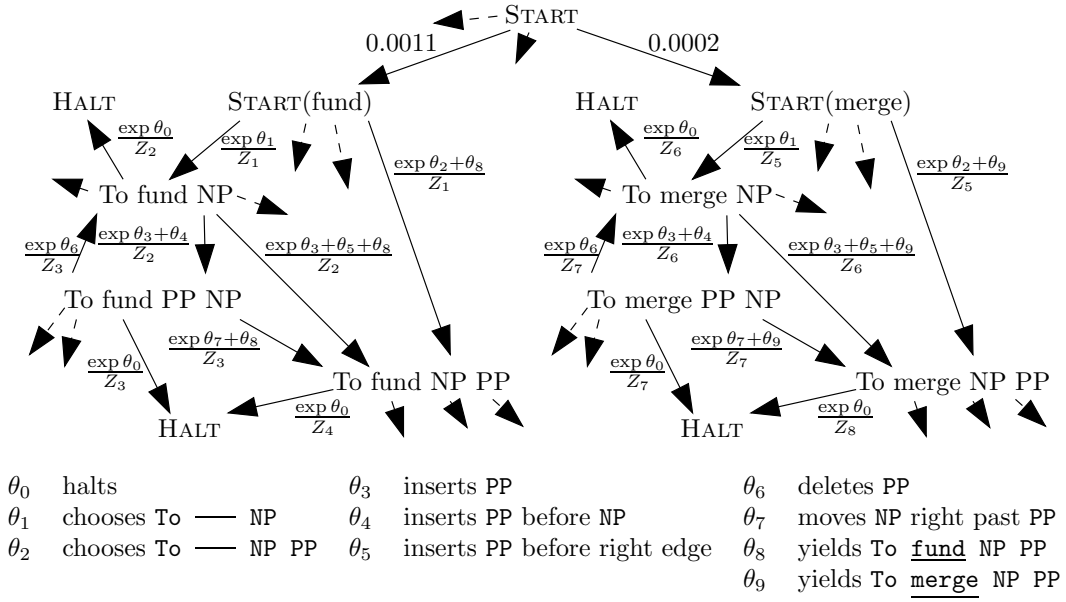

Figure 1.3: How the arc probabilities in Fig. 1.2 were determined from feature weights $\vec{\theta}$. The Z values are chosen so that the arcs leaving each vertex have total probability 1.

one another; and if θ_3 is positive, then PP-inserting transformations have an advantage in this competition.

If PP-insertion is common among the frequent words where we have a chance to observe it, we will learn that θ_3 is large in this language. This generalization also raises the probabilities of PP-insertion transformations that have never been attested, simply because these transformations also have θ_3. So it affects our estimates of lexical entries even for poorly-observed words.

As another example, the weight θ_9 is known as a "per-event" weight. It appears on all and only the arcs to S → To merge NP PP. Large θ_9 says that that lexical entry—*and its descendants in the graph* (§1.4.1)— have higher probabilities than one would otherwise expect. Put another way, this single parameter value says that merge is listed as a transitive verb with a role for a PP, and (other things equal) enjoys all the ordinary privileges of such verbs.

To flesh out the first paragraph of this section: The vertex labels, graph topology, and arc parameters of Fig. 1.3 are language-independent. More bluntly, this figure constitutes

universal grammar. Learning a specific language's syntax is now just a matter of learning the weight vector $\vec{\theta}$. In other words, it is matter of learning what features of a linguistically possible transformation make it likely in the language.

1.4.3 The Prior

"With three parameters I can fit an elephant," wrote Lord Kelvin. There is always a danger of having so many parameters that one overfits the training data. On the other hand, having too few parameters is also dangerous, since elephants are sometimes the rule rather than the exception. Our approach is to permit very many degrees of freedom, but to use a prior to discourage their indiscriminate use.

While our use of feature weights (§1.4.2) may reduce the number of parameters, it still leaves too many parameters to estimate them freely from the available data. Indeed, the use of per-event weights such as θ_9 means that there are usually more parameters than training data—an underdetermined model.

Simply setting the weights to maximize the likelihood of training data would therefore lead to serious overfitting. Rather than smooth the data transformationally, it would just yield the maximum-likelihood multinomial, in which the estimated probability of a lexical entry is simply proportional to the number of times it was observed (perhaps zero). This is exactly the situation we were trying to avoid in §1.2.1.2.

The antidote is to assume a prior distribution over the parameters. We can then learn by maximizing not the likelihood, but the joint probability of the training data and parameters:

$$\underbrace{\Pr(\text{observed lexical entries} \mid \theta)}_{\substack{\text{"likelihood"} \\ = \Pr(\text{training data}|\text{grammar})}} \cdot \underbrace{\Pr(\theta)}_{\substack{\text{"prior"} \\ = \Pr(\text{grammar})}} \tag{1.2}$$

This is simply an instantiation of equation (1.1).

What prior $\Pr(\theta)$ will we use? If a feature has weight 0, it has no effect on the probabilities of arcs that bear it. We expect *a priori* that most features are innocuous in most languages: $\theta_i \sim N(0, \sigma^2)$, so crosslinguistically feature weights tend to be close to 0. It takes a fair amount of evidence to convince us that in the language we are learning, the

unexpected is true and a particular feature θ_i has high weight.

If PP-insertion before NP has a specially high or low probability, the parameter θ_4 lets us model that probability accurately. But without such evidence we will try to respect our prior belief that $\theta_4 \approx 0$, meaning that PP is about as likely to insert before NP as anywhere else. Similarly, without a lot of evidence that S → To merge NP PP has a special probability, we will tend to assume $\theta_9 \approx 0$, meaning that Pr(S → To merge NP PP) is predicted by the same transformational processes as Pr(S → To fund NP PP.

In practice, the evidence about S → To merge NP PP is rarely either plentiful or absent. So rather than choose the empirically observed or the transformationally predicted probability, we must interpolate between them to maximize equation (1.2). This is the transformational smoothing effect.

It is really the prior that encourages transformational smoothing to find generalizations. Since the model is underdetermined, there are multiple ways to account for the training data with parameters. If during learning, S → To fund NP PP and S → To merge NP PP are both observed more often in training data than predicted by the model, then one has two alternatives:

- Raise their probabilities separately by increasing the per-event weights θ_8, θ_9. This makes two stipulations.
- Raise their probabilities together by raising θ_4 (or perhaps even θ_3). This expresses a broader generalization about PP insertion.

Both alternatives equally help the likelihood term of equation (1.2). So the latter is preferred, other things equal, because it hurts the prior term less.

1.5 Results

1.5.1 Algorithmic Results

In Fig. 1.2, the arc probabilities define a recurrence relation in which each lexical entry's probability is a linear function of its parents' probabilities. One can obtain the distribution

over entries by solving this linear system.[13] Using sparse matrix methods, this can be done in $O(n|F|)$ time and $O(n)$ space,[14] where n is the number of vertices and $|F|$ is the total number of feature weight tokens in Fig. 1.3 (the "size" of the model).

One can improve the objective function equation (1.2) by adjusting θ, using gradient-based optimization or Expectation-Maximization. One step of either method—computing the gradient or the expected traversals of each feature—corresponds to solving a linear system, again in $O(n|F|)$ time and $O(n)$ space.

If n is large or infinite it is necessary to use approximations for all these methods (solving the model, computing the gradient, and running EM). One can consider only random walks of some length T or less. Then the runtime and space become $O(T|F|_T)$ and $O(nT + k)$, where $|F|_T$ is the size of the part of the model that can be reached from START by paths of length $\leq T$, and k is the number of feature types. The algorithms are closely related to propagation and back-propagation through time in "unrolled" recurrent neural networks (§8.5.4), with some additional optimizations. Improving on this, a more flexible class of algorithms is derived from relaxation methods.

Because our transformation model of the lexicon has a particular (not uncommon) form, some additional model-specific optimizations are possible. Most importantly, one can exploit the fact that the subgraphs corresponding to different headwords have isomorphic topology and similar probabilities.

1.5.2 Empirical Results on the Test Set

§6.7 compares several modeling techniques, using comparable features, according to how well they predict $\texttt{S} \rightarrow \cdots$ lexical entries in the Penn Treebank. A simple transformation model, estimated without great care in the smoothing parameters, reduces test-set perplexity by 20% over the best model from the literature. This reduction is comparable to doubling the amount of training data.

The reduction is somewhat greater on a smaller training set, highlighting the model's

[13] If the transformation graph is acyclic, the solution is considerably faster. However, disallowing cycles would force the model designer to decide in advance whether it is intransitives that are derived from transitives, or vice versa, rather than learning this from data.

[14] Alternatively, $O(|F| + n|P|)$ time and $O(|P|)$ space, where $|P|$ is the number of arcs in Fig. 1.3.

ability to generalize from sparse data, and can be increased somewhat by model averaging.

The improvement due to transformation modeling seems to stem from three sources: fitting the training data more exactly, generalizing better, and weighting the backoff evidence in a Bayesian fashion.

These experiments using comparable features show that transformational smoothing can beat other methods even without wiring any real linguistics into the transformation graph. In future work, we hope to increase its advantage—especially by adding "interesting" transformations like extraction, passivization, and tense formation, which the other methods have little hope of modeling—and translating this advantage into gains in parsing performance, particularly on limited amounts of data.

1.5.3 Empirical Results on the Example

One can get a qualitative sense of how transformational smoothing performs, relative to the best model from previous literature, by comparing their assessments of the lexical entries in Tables 1.3 and 1.4.

- The average entry in the table is 1.44 times as likely as before (geometric mean). Equivalently, the set of table entries has 31% lower perplexity.[15]

- The transformation model fits the training data more closely (because it can model exceptions). It assigns 53% lower perplexity to the training set in Table 1.1. Moreover, on these test data it separates observed and unobserved entries, assigning probabilities > 0.05 to the former and < 0.025 to the latter. Such a separation is not observed for the competing model.

- The transformation model also assigns higher probabilities to the unobserved entries in this table, reducing the perplexity of that set by 62%.

- The transformation model wastes considerably less probability on the "other" line—the multitude of frames that did not appear with any of these words. Indeed, it allocates more than 50% as much *total* probability to the entries in the table.

[15] Of course, this is not a typical test set: common and uncommon entries alike appear exactly once. As a result it has twice the perplexity of the real test set.

- One can observe transformational generalizations at work between S → NP —— NP . and S → NP —— NP PP . (the frames in rows 5–6 of Table 1.4). These frames receive probabilities of about 0.0076 and 0.0026 when neither has been observed with a word (columns 4–6). But columns 1–2 show that observing either one increases the probability of the other.

  To be precise, observing S → NP encourage NP PP . increases its estimated probability by quite a lot. Thanks to a Delete-PP transformation, S → NP encourage NP . then automatically increases by 18% as much. There is also an Insert-PP transformation giving the converse effect: twice observing S → NP question NP . increases its estimated probability; then thanks to Insert-PP, S → NP question NP PP . automatically increases by about 4% as much.[16]

- Transformational smoothing smooths the probabilities of observed entries, not just unobserved ones. In rows 3–4, as in rows 5–6, the expected probabilities of entries with and without PP are in about 3:1 ratio. But S → To AdvP remove NP and S → To AdvP remove NP PP have been *observed* in 1:1 ratio. The estimates in the table interpolate between these: they have the compromise ratio of 1.6 : 1.

The first few bullet points above highlight the fact that the competing model of lexical entries, a "bigram model," is relatively indiscriminate as to where it throws probability. §6.6.4 remedies this by improving the competing model, instead using maximum likelihood with backoff from the headword and backoff to the bigram model.

[16] It is surprising, but consistently the case, that deletions emerge with higher probability than insertions. There are two possible explanations. One is that this untraditional result should be regarded as correct, on the grounds that most dependents fill semantic roles and are more like arguments than adjuncts. (Then most variation among frames could be regarded as arising from deleting subcategorized roles rather than inserting adjuncts.) The other explanation is that it is harder to *learn* insertions. For a given lexical entry, a few deletion transformations are competing with literally hundreds of insertion transformations that want to insert various nonterminals at various positions in the entry. This necessarily leads to a generic bias against insertions, which the model must learn to overcome for particular kinds of insertions.

In principle, a transformation model will perform abduction (just like a Bayes net), reasoning backwards from effects to their causes. Thus, even if Insert-PP has probability 0, an observation of f_2 = S → NP question NP . is still weak indirect evidence of f_1 = S → NP question NP PP . since the f_1 is a possible precursor of the observation f_2 under Delete-PP. However, the model used in the experiments of this thesis does not have enough free parameters to effectively abduce lexical entries like f_1 that were never observed. (As §6.4.3 explains, only other observed entries are permitted to have exceptional probabilities that could be tuned to account for f_2.)

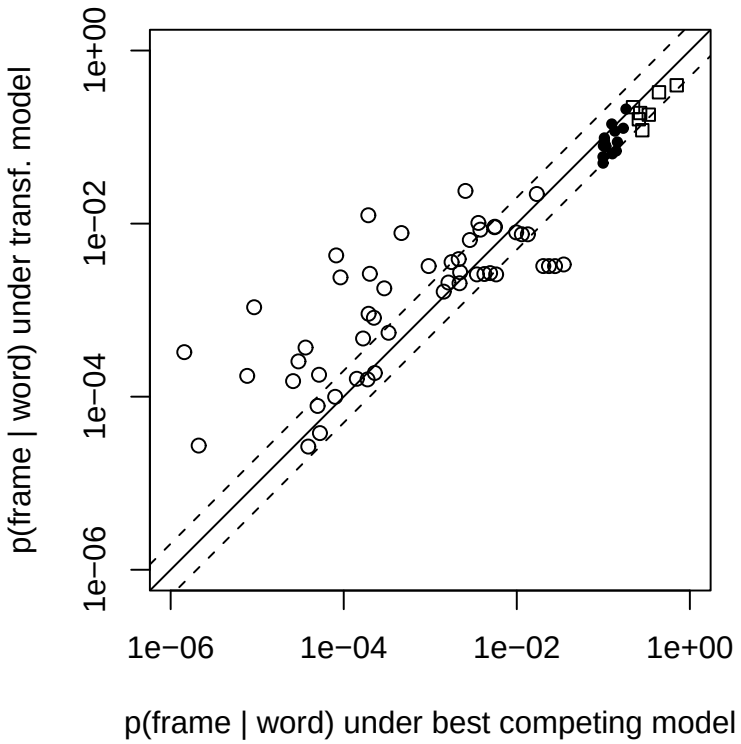

Figure 1.4: All probabilities in Table 1.4, plotted against the corresponding probabilities from the best competing model (§6.6.4—an improvement on the model discussed in most of §1.5.3). Lexical entries receiving a higher probability from the transformation model fall above the middle diagonal. The dashed outer diagonals mark a factor-of-2 difference between the two models. The plotting symbol indicates the number of training observations: $\circ = 0$, $\bullet = 1$, $\square \geq 2$.

In the final results (§6.7.1), that improved model was the best of the transformation-free models. Fig. 1.4 shows that it allocates slightly more probability to training data, but that the transformation model quite consistently assigns more probability to the novel "test" data in our example. The exceptions (slightly below the diagonal) are novel entries for the three frames that end in periods.[17] The biggest winners under the transformation model are some of the novel entries for `fund`, which was observed with the fewest and most similar frames in training data; the transformation model extrapolated more aggressively from these to the related frames in the table.[18]

1.6 Structure of the Thesis

To a large extent, the remaining chapters can be read independently, since they lay out different facets of the work: linguistics, statistics, algorithms, parsing, and experimental evaluation. However, the chapters are arranged in a natural sequence and sometimes refer to one another.

This introductory **Chapter 1** used an example (Table 1.1) to show how four problems converge on a single approach: the engineering problem of generalizing better from data when training a statistical parser (§1.2.1), the linguistic problem of attaching probabilities to lexical redundancy rules and exceptions (§1.2.2), the statistical problem of modeling positive covariance in the probabilities of mutually exclusive events (§1.2.3), and the learning problem of stretching a grammar to interpret new utterances (§1.2.4). Taken together, these problems motivate a new approach to grammar cost (§1.3) that can be formalized as a new kind of statistical model (§1.4). The approach is algorithmically manageable (§1.5.1) and yields an empirical improvement over work from the literature (§§1.5.2–1.5.3). So much for the overview.

Chapter 2 presents the linguistic idea of the thesis. It begins by describing lexicalized theories of syntax (§2.1) and motivating them (§2.2). Such a theory describes a language

[17] The transformation model did not learn a high probability for period-insertion. Even in general it is tricky to learn high rates for insertions (footnote 16), and in this case, the model did not have access to the "tensed verb" feature on which period-insertion should be conditioned (see footnote 4 on p. 10).

[18] Part of the reason is that the backoff technique (§6.6.2) in the competing model did not back off very far from the observed data for `fund`, taking the lack of 1's in `fund`'s column in Table 1.1 as a sign that `fund` had been well observed.

on four levels (§2.1.5): (1) strings, (2) a lexicon of "chunks" of surface structure, (3) a language-specific deep structure that interrelates the "chunks," and (4) a Universal Grammar that constrains and influences the form of the deep structure. Statistics is arguably part of humans' knowledge of language, and can be used to enrich each level (§2.3).[19] In particular, chunks have probabilities (§2.3.4), and we propose that deep structure concerns *correlated* probabilities and can be modeled by stochastic transformations on the lexical entries (§2.3.5). Since real grammars contain idiosyncrasies, these correlations are often imperfect; but the imperfections (which are matters of degree) can be modeled as well, and in such a way that a learning algorithm will attempt to characterize the lexicon in terms of the strongest correlations and the mildest exceptions possible (§2.3.6).

Chapter 2 continues by outlining the particular lexicalized theory to be used in the experiments. In this simple theory, lexical entries are "flat" context-free rules (§2.4.1). Transformations act to insert, delete, replace, or permute the nonterminals in a rule (§2.4.2). Such transformations serve to modify semantic argument structure, and their effects are evident in the training data (§2.4.2.1). Other linguistic transformations could also be useful (§2.4.3). The chapter closes by reviewing related work on lexical organization (§2.5.1), lexicon smoothing (§2.5.2), the extraction of subcategorization frames from text (§2.5.3), the modeling of optionality in subcategorization frames (§2.5.4), translation and summarization based on local syntactic transformations (§2.5.5), and approaches to grammar learning based on edit distance (§2.5.6) or priors on grammars (§2.5.7).

Chapter 3 presents the statistical framework (without reference to the linguistic application), just as Chapter 2 presented the linguistic framework. It begins with a review of probability notation and Bayesian smoothing (§3.1). Then it proposes a new class of statistical models (§3.2). These "transformation models" are related to random walks (§3.3). The chapter continues with practical advice about how to compute with transformation models (§3.4), how to design priors for them (§3.5), and how to design models that can capture exceptions (§3.6).

Transformation models are designed to model probability distributions in which "related" events have correlated probabilities (§3.7.1). §3.7.2 sketches the application to

[19] §2.3.7 meditates on the consequences for grammaticality judgments.

smoothing of a syntactic lexicon. §3.8 analyzes a very simple transformation model in detail, showing that the prior prefers to capture generalizations (§3.8.4) but that the evidence can motivate exceptions (§3.8.5). Estimating the parameters of the model yields a smoothed distribution (§3.8.6). Qualitatively, frequent events in the training data influence the smoothing of infrequent events more than vice-versa, but (fortunately) not in direct proportion to their frequency; rather, they are influential to the extent that their probabilities are nailed down by the evidence (§3.8.7). Parameter estimation can be difficult, since while the prior is a unimodal distribution over distributions (§3.8.8), the posterior unfortunately has local maxima. The chapter closes by proposing a variation, "perturbed" models (§3.9, which model exceptions in a slightly different way (§3.9.3): such models introduce extra parameters that correspond to flow multipliers in generalized network flow problems.

Chapter 4 is the computational heart of the thesis. It presents parameter estimation methods for arbitrary transformation models. §4.1 defines transformation models again, this time with a concise matrix notation that is used throughout the chapter. It also restates the objective function to be maximized during training (§4.1.4) and the evaluation function for testing (§4.1.5).

It is possible to approximate the objective function by a simple and flexible relaxation algorithm (§4.2), and to find the gradient of this approximation by "back-relaxation" (§4.3), for which a correctness proof is given (§4.3.4). The method can be extended without much difficulty to handle perturbed models as well (§4.4).

Chapter 4 closes with some ways to make relaxation and back-relaxation more efficient. Many arcs in the transformation graph do not affect the computation much, or at all, and can be ignored for speed (§4.5.2). Another important optimization applies to a common class of transformation models that have repetitive topology (§4.5.3). For example, most headwords agree on the probabilities of most syntactic transformations, so rather than repeat all computations for each headword, it is only necessary to propagate a few differences from a template.

Chapter 5 places the work in the context of statistical natural-language parsing. It sketches how one might practically use a transformationally smoothed syntactic lexicon

in a parsing system. While transformational smoothing could be used with any lexicalized theory of grammar, the framework assumed here is lexicalized, flattened context-free grammar (§2.4.1).[20] §§5.1–5.3 formally define this framework. In particular, they define the probabilities of syntax trees in terms of probabilities in the lexicon, using a standard generative model in which nonterminals are recursively expanded into full lexical entries. Each expansion can be broken down into two steps (§5.4.1), both of which can exploit the smoothed lexicon. The first step is a contextually appropriate choice of headword for the nonterminal; since this choice must be compatible with the nonterminal (representing the headword's maximal projection) as well as with the governing lexical item, it is partly a syntactic choice, and the syntactic lexicon can help determine its probability (§5.4.3). The second step is to choose a sequence of dependent nonterminals for that headword; specifying the probability of this choice is the main job of the syntactic lexicon (§5.4.2). The two steps can be cleanly combined under certain Naive-Bayes assumptions (§5.4.5).

The rest of Chapter 5 is "armchair linguistics," conducted in the interest of concreteness and with an eye toward future experiments. It argues that with appropriate tricks, the lexicalized, flattened context-free approach would in principle be good enough to accommodate and exploit a variety of linguistic devices: category-changing transformations (§5.4.4), morphemes (§5.5.1.1), semantic clusters of words (§5.5.1.2), word senses (§5.5.1.3), long-distance movement (§5.5.3), syntactic as well as semantic heads (§§5.5.4.1–5.5.4.2), feature agreement and checking (§5.5.4.3), dependencies on more than one word (§5.5.4.4), thematic roles (§5.5.4.5), and dependence on greater amounts of context (§5.5.5).

Chapter 6 contains the experimental evaluation. Following the generative model of syntax trees in the previous chapter, the measure used for comparison is the perplexity of the right-hand-side of a context free rule (§6.1). The training and test data were lexical entries extracted from the Penn Treebank (§6.2); there were substantial amounts of novelty in the test data (§6.3).

The chapter continues by specifying the details of the transformation model (§6.4) and the methods for estimating its parameters (§6.5) that were used in the experiments. The

[20] That is, each context-free rule (lexical entry) rewrites a maximal projection in one step as a headword together with all its dependents. The resulting trees are similar in spirit to dependency trees, except that the subtrees are labeled with nonterminals rather than semantic roles.

competing models include a few from the literature (§6.6.1), with appropriate smoothing (§6.6.2), and tested using both flat and non-flat lexical entries (§6.6.3). Some new hybrid models are also considered (§6.6.4).

The experimental results examine the competing models and the overall perplexity reduction (§6.7.1), as well as the consistency (§6.7.2), difficulty (§6.7.3), and training-size sensitivity (§6.7.4) of this reduction. It used various additional experiments to investigate the contributions of memorizing the training examples (§6.7.1), using a Bayesian scheme to weight backoff evidence (§6.7.5), and generalizing better (§6.7.6). Finally, it briefly looks at the time course of learning and the distribution of learned weights (§6.7.7).

The experiments having been described, **Chapter 7** picks up where Chapter 3 left off, and makes some further remarks on transformational smoothing. It draws connections from transformation models to Markov processes (§7.1.1), finite-state machines (§7.1.2), recurrent neural networks (§7.1.3), graphical models (§7.1.4), and Bayesian backoff (§7.1.5). It also gives some non-linguistic applications of transformation models (§7.2). Finally, it offers a number of possible variations on the formalism. One can modify the form of the model (§7.3.1) or the form of the prior (§7.3.2). In fact, a simple modification to the prior yields Zipfian (power-law) behavior of a sort often observed in languages (§7.3.3). One can also interpret the model parameters differently so that the transformational process is blessed with "lookahead" (§7.3.4): there are at least three ways to do this, including global normalization of path probabilities (§7.3.4.2) and a "transformational circuit" approach inspired by current flow in electrical networks (§7.3.4.3).

Chapter 8 similarly augments Chapter 4, offering some additional algorithms for estimating the parameters of transformation models. §8.1 shows that the objective function and its gradient can be computed in closed form if desired (as opposed to the converging relaxation algorithm of Chapter 4). Moreover, the computations can be made to exploit the sparsity of the transformation graph. An alternative to gradient descent is to use Expectation-Maximization (§8.2), where the hidden variables are the paths in the transformation graph that generated the observed events. The M step of EM uses these paths to estimate the parameters of a log-linear distribution (§8.2.1), using a standard method like Improved Iterative Scaling (§8.2.2). The E step of EM reconstructs the paths for use

by the M step: it is conceptually related to the forward-backward algorithm, and consists of solving a single sparse linear system that is defined by the training dataset and current parameters (§8.2.3). Some variations on EM are discussed in §8.2.4, including an algorithm for the Viterbi approximation and confidence-weighted approaches, and §8.3 shows that the central computation of EM can be efficiently approximated using the very same back-relaxation algorithm proposed in Chapter 4.

§8.4 discusses when, why and how one might wish to renormalize the probability distribution $\vec{p}$ defined by a transformation model. The chapter closes with somewhat tedious generalizations of two of the ideas from Chapter 4. The (back-)relaxation algorithm can be replaced with a pedagogically simpler (back-)propagation algorithm (§8.5) that is related to back-propagation in recurrent neural networks. The "template" approach of §4.5.3 can also be generalized to a variety of other circumstances (§8.6), for both exact and approximate algorithms.

Chapter 9 offers some brief conclusions and a discussion of future work.