

Jason M. Eisner [\[March 14, 2007\]](#)

Statement of Research Accomplishments and Goals

[\[Past papers are hyperlinked in red.\]](#)

My research tries to improve the fundamental techniques of natural language processing (NLP). I design core algorithms, learning procedures, and software that can be applied to help engineer solutions across many practical NLP tasks.

The end goal of NLP is to give computers some ability to read, hear, and produce language as a human would. Then computers can help us manage the vast quantities of text and speech (in many languages and genres) that reach us or are available to us electronically. The bulk of human knowledge and opinion is expressed as natural language; the bulk of human activity is mediated by written or spoken communication. Thanks to the Internet and a proliferation of personal electronic devices, more and more of this knowledge and communication is at least temporarily available in digital form.

While some of my work has potential applications outside NLP (especially to other one-dimensional data such as genomics, time series, music, and visual contours or trajectories), I am particularly interested in the structure of human language. Thus, I often design **algorithms** for applying and learning **statistical models** that exploit **linguistic structure to improve performance on real language data**. One recurring algorithmic theme has been **dynamic programming**.

Parsing: Automated sentence diagramming, or parsing, is a key step in obtaining a sentence's meaning or translation. Leading research systems for question answering, information extraction, and machine translation now rely on parsing algorithms, rather than using cruder substitutes.

I devised fundamental, widely-used dynamic programming algorithms for **dependency grammars**, **combinatory categorial grammars**, and **lexicalized CFGs and TAGs**. They allow parsing to remain asymptotically efficient when grammar nonterminals are enriched to record arbitrary sequences of gaps [3] or lexical headwords [4,6,7,8,9]. Recently I showed that they can also be modified to obtain accurate, **linear-time partial parsers** [10].

In **statistical parsing**, I was one of the first researchers to **model lexical dependencies** among headwords [1,2], the first to model **second-order effects** among sister dependents [4,5], and the first to use a **generative lexicalized model** [4,5], which I showed to beat non-generative options. That successful model had the top accuracy at the time (equalling Collins 1996) and initiated a 5-year era dominated by generative, lexicalized statistical parsing. The most accurate parser today (McDonald 2006) continues to use the algorithm of [4,9] for English and other projective languages.

Recently, I have shown how various **parsing algorithms are interrelated by formal transformations** that appear widely applicable [35]. I continue to explore new uses of such transformations.

Good parsers are unfortunately not available for all languages, or even for most genres of English text, owing to a shortage of training examples (i.e., hand-produced sample parses). One of my major new research goals is to address this problem, **automatically learning parsers for new languages or genres** by exploiting unannotated text (monolingual and bilingual) together with the knowledge inherent in existing parsers or treebanks.

- [1] [A Probabilistic Parser and Its Application](#) (1992), with Mark Jones
- [2] [A Probabilistic Parser Applied to Software Testing Documents](#) (1992), with Mark Jones
- [3] [Efficient Normal-Form Parsing for Combinatory Categorial Grammar](#) (1996)
- [4] [Three New Probabilistic Models for Dependency Parsing: An Exploration](#) (1996)
- [5] [An Empirical Comparison of Probability Models for Dependency Grammar](#) (1996)
- [6] [Bilexical Grammars and a Cubic-Time Probabilistic Parser](#) (1997)

- [7] [Efficient Parsing for Bilexical Context-Free Grammars and Head Automaton Grammars](#) (1999), with [Giorgio Satta](#)
- [8] [A Faster Parsing Algorithm for Lexicalized Tree-Adjoining Grammars](#) (2000), with [Giorgio Satta](#)
- [9] [Bilexical Grammars and Their Cubic-Time Parsing Algorithms](#) (2000)
- [10] [Parsing with Soft and Hard Constraints on Dependency Length](#) (2005), with [Noah Smith](#)

Grammar induction and learning: As noted above, statistical parsing raises the question of where to get the statistical grammars. My students and I have developed several state-of-the-art approaches.

The “obvious” method is the Expectation-Maximization (EM) algorithm, but it is well known to get hopelessly trapped in local optima. My students and I have demonstrated the practical benefit of **various “annealing” techniques** [17,23,24,25] that start with a simpler optimization problem and gradually morph it into the desired one. In particular, initially biasing toward *local* syntactic structure [10] has obtained the **best known results in unsupervised dependency grammar induction** across several languages [24], resulting in parse structures that are good enough to be potentially useful. We have separately used annealing techniques to **refine grammar nonterminals** [25] and to **minimize task-specific error** in parsing and machine translation [23].

Our other major improvement over EM is **contrastive estimation** [18,19], which modifies EM’s problematic objective function (likelihood) to use implicit negative evidence. The new objective makes it possible to discover both part-of-speech tags and dependency relations where EM famously fails. It is also more efficient to compute for general log-linear models.

For finite-state grammars, I introduced the general EM algorithm for **training parametrically weighted regular expressions and finite-state machines** [12,13], generalizing the forward-backward algorithm [14].

The above techniques can be applied to unsupervised or semi-supervised learning problems. However, unsupervised learning opportunities remain even in the supposedly “supervised” case of Treebank data, where grammar rules are directly observed. I have developed a statistical smoothing method, **transformational smoothing** [11,15,16], that models how the probabilities of deeply related rules tend to covary. It discovers this linguistic **deep structure** without supervision. It also models cross-lexical variation and sharing among lexicalized rules; this could alternatively be done by **generalizing latent Dirichlet allocation** to finite-state models [22].

Unsupervised learning is of course not the only practical scenario. I have lately been working on new techniques to get higher performance with only small amounts of supervision. A particular goal is to bootstrap high-performance parsers within and across languages; another (interlocking) goal is active learning of morphology. This work involves new kinds of models and learning procedures.

Once we are considering using some supervision, non-traditional approaches may be able to **reduce the cost of linguistic annotation**. I have been exploring this possibility starting with simpler tasks. For instance, annotators can highlight contextual clues to their decisions (“**rationales**,” which provide valuable side information to a machine learning algorithm [28]). Unsupervised sniff tests can replace supervision for purposes of model selection (“**strapping**,” which outperforms human intuition in the choice of seeds for bootstrapping [20]). The behavior of an unsupervised algorithm on some problem can be tuned using available supervised data for a *different* instance of the same problem (“**cross-instance tuning**,” a kind of “learning how to learn” [20,26,27]). Another opportunity for cheap “supervision” is to incorporate some **knowledge in the task domain**. For example, most work on grammar learning makes surprisingly little use of straightforward knowledge that a 1-year-old might bring to that task, a research opportunity that [18,19,24] only began to scratch.

- [11] [Smoothing a Probabilistic Lexicon Via Syntactic Transformations](#) (2001)
- [12] [Expectation Semirings: Flexible EM for Finite-State Transducers](#) (2001)
- [13] [Parameter Estimation for Probabilistic Finite-State Transducers](#) (2002)
- [14] [An Interactive Spreadsheet for Teaching the Forward-Backward Algorithm](#) (2002)
- [15] [Transformational Priors Over Grammars](#) (2002)
- [16] [Discovering Syntactic Deep Structure via Bayesian Statistics](#) (2002)
- [17] [Annealing Techniques for Unsupervised Statistical Language Learning](#) (2004), with Noah Smith
- [18] [Contrastive Estimation: Training Log-Linear Models on Unlabeled Data](#) (2005), with Noah Smith
- [19] [Guiding Unsupervised Grammar Induction Using Contrastive Estimation](#) (2005), with Noah Smith
- [20] [Bootstrapping Without the Boot](#) (2005), with Damianos Karakos
- [21] [Unsupervised Classification via Decision Trees: An Information-Theoretic Perspective](#) (2005), with Karakos et al.
- [22] [Finite-State Dirichlet Allocation: Learned Priors on Finite-State Models](#) (2006), with Jia Cui
- [23] [Minimum-Risk Annealing for Training Log-Linear Models](#) (2006), with David Smith
- [24] [Annealing Structural Bias in Multilingual Weighted Grammar Induction](#) (2006), with Noah Smith
- [25] [Better Informed Training of Latent Syntactic Features](#) (2006), with Markus Dreyer
- [26] [Iterative Denoising Using Jensen-Renyí Divergences with an Application to Unsupervised Document Categorization](#) (2007), with Karakos et al.
- [27] [Cross-Instance Tuning of Unsupervised Document Clustering Algorithms](#) (2007), with Karakos et al.
- [28] [Using “Annotator Rationales” to Improve Machine Learning for Text Categorization](#) (2007), with Omar Zaidan and Christine Piatko

Machine translation: Extending parsing techniques to MT, one would like to jointly model the syntactic structure of an English sentence and its translation. I have designed flexible models [29,30,31] that can **handle imprecise (“free”) translations**, which are often insufficiently parallel to be captured by synchronous CFGs (e.g. ITGs).

Very recently, I pointed out a much less obvious MT-parsing connection. Parsing can be used to attack the NP-hard problem of reordering the source-language words in an optimal way before translation. I developed powerful **iterated local search** algorithms for such NP-hard permutation problems (as well as classical NP-hard problems like the TSP) [32]. The algorithms borrow various parsing tricks in order to explore **exponentially large local neighborhoods** in polytime. We are now exploring the empirical performance of this new approach, both for existing models (e.g., phrase-based translation) and planned new models (with novel types of reordering costs permitted by our algorithms).

Multilingual data is also used in some of my other recent work and that of my students [10,20,23,24; 64,65,66].

- [29] [Learning Non-Isomorphic Tree Mappings for Machine Translation](#) (2003)
- [30] [Natural Language Generation in the Context of Machine Translation](#) (2004), with Hajič et al.
- [31] [Quasi-Synchronous Grammars: Alignment by Soft Projection of Syntactic Dependencies](#) (2006), with David Smith
- [32] [Local Search with Very Large-Scale Neighborhoods for Optimal Permutations in Machine Translation](#) (2006), with Roy Tromble

The Dyna language (www.dyna.org): Progress in NLP can be slow. Many ideas are stalled, or never pursued, because it is so time-consuming to build correct systems that are efficient enough to handle real-world test data.

My **new programming language**, Dyna [33,34], makes it much faster and easier to build such systems and run full-scale experiments. It encapsulates and generalizes many years of parser-building experience. In Dyna, one can **express dynamic programs** and related computational schemes (e.g.,

[11,37]) as small sets of “Horn equations.” Their efficiency can be further improved by **program transformations** that derive new algorithms from old ones [35,34].

Our prototype **Dyna compiler** turns these short declarative specifications into efficient C++ classes, which instantiate generalized algorithms (more general than previous ones) for **update propagation** and **automatic differentiation** [34,11]. To help you understand and improve systems written in Dyna, we have also built (and continue to develop) a **visual debugger**, Dynasty, that does dynamic hypergraph layout and lets you explore huge parse forests and proof forests [36].

We have used prototypes of these tools in at least 15 published papers and 2 undergraduate classes. We are now designing many improvements to their expressiveness and efficiency. Expressiveness is undergirded by our formalism of **weighted logic programming** [35]. On the efficiency front, our goal is to permit manual exploration—and later automatic exploration—of a **large space of possible implementations** of any given algorithm. Options in this space are selected by adding declarations to the Dyna program. They include program transformations as mentioned above; strategies for memory layout, indexing, type specialization, update propagation, and memoization; and heuristic policies for prioritization and pruning, which we are optimizing by machine learning on actual workloads.

- [33] **Dyna: A Declarative Language for Implementing Dynamic Programs** (2004),
with Eric Goldlust and Noah Smith
- [34] **Compiling Comp Ling: Weighted Dynamic Programming and the Dyna Language** (2005),
with Eric Goldlust and Noah Smith
- [35] **Program Transformations for Optimization of Parsing Algorithms and Other Weighted Logic Programs** (2006), with John Blatz
- [36] **Visual Navigation Through Large Directed Graphs and Hypergraphs** (2006), with several undergraduates
- [37] **Dynamical-Systems Behavior in Recurrent and Non-Recurrent Connectionist Nets** (1990)

Finite-state machines: A core NLP technology is provided by weighted finite-state automata and transducers, which can be used to represent and combine many useful functions on strings, as well as data such as dictionaries and speech recognition lattices.

I published the most general algorithms for both **training** [12,13] and **minimization** [38] of weighted finite-state machines, as well as some interesting theoretical limitations on minimization [38] and on treating multi-tape machines as infinite relational databases [40,41]. I have also developed **new finite-state approaches** in several applied contexts, including machine translation (local constraints) [32], information extraction (efficient global constraints) [42], user interfaces [40], cryptanalysis [43], and computational phonology [45,50,51].

I am presently studying **Markov random fields over string-valued random variables**, where different strings are related by finite-state potential functions. One application is joint modeling of morphological paradigms. We are also designing a flexible, trainable, efficient **finite-state toolkit** to be built in Dyna [34,35,36].

- [38] **Simpler and More General Minimization for Weighted Finite-State Automata** (2003)
- [39] **Radiology Report Entry with Automatic Phrase Completion Driven by Language Modeling** (2004),
with John Eng
- [40] **A Note on Join and Auto-Intersection of n -ary Rational Relations** (2004), with Kempé et al.
- [41] **A Class of Rational n -WFSM Auto-Intersections** (2005), with Kempé et al.
- [42] **A Fast Finite-State Relaxation Method for Enforcing Global Constraints on Sequence Decoding** (2006), with Roy Tromble
- [43] **A Natural-Language Approach to Automated Cryptanalysis of Two-Time Pads** (2006),
with Mason et al.

Computational phonology and finite-state methods: I proposed perhaps the leading **formalization** [44,45,47] of allowable constraints and representations in Optimality Theory (the main approach to phonology since the early 1990's [48]). This formalization, Primitive OT, permits **finite-state computation** [45] and has led to a new, **confirmed cross-linguistic prediction** [46]. I have also proposed a **modification to Optimality Theory** that further improves its computational and linguistic properties [50]. I have published key algorithms and complexity theorems on all three of the major computational problems raised by Optimality Theory: **generation** [45,50,51], **comprehension** [50,51], and **learning** [49]. I have recently begun collaborative discussions on statistical learning of constraint grammars.

- [44] [What Constraints Should OT Allow?](#) (1997)
- [45] [Efficient Generation in Primitive Optimality Theory](#) (1997)
- [46] [FootForm Decomposed: Using primitive constraints in OT](#) (1997)
- [47] [Doing OT in a Straitjacket](#) (1999)
- [48] [Review of *Optimality Theory* by René Kager](#) (2000)
- [49] [Easy and Hard Constraint Ranking in Optimality Theory: Algorithms and Complexity](#) (2000)
- [50] [Directional Constraint Evaluation in Optimality Theory](#) (2000)
- [51] [Comprehension and Compilation in Optimality Theory](#) (2002)

Miscellaneous research: I have done occasional work on linguistic semantics [53], information extraction [42,54], text compression [59], collaborative filtering and online commerce [55,56,57,58], and even practical voting systems [52]. I am generally interested in modeling interesting domains (not necessarily limited to natural language).

- [52] [Indirect STV Election: A Voting System for South Africa](#) (1991)
- [53] [‘All’-less in Wonderland? Revisiting *any*](#) (1995)
- [54] [Description of the University of Pennsylvania entry in the MUC-6 competition](#) (1995)
- [55] [System for Generation of Object Profiles for a System for Customized Electronic Identification of Desirable Objects](#) (1995), [with Herz et al.](#)
- [56] [System for Generation of User Profiles for a System for Customized Electronic Identification of Desirable Objects](#) (1995), [with Herz et al.](#)
- [57] [Pseudonymous Server for System for Customized Electronic Identification of Desirable Objects](#) (1995), [with Herz et al.](#)
- [58] [System for the Automatic Determination of Customized Prices and Promotions](#) (1996), [with Herz et al.](#)
- [59] [A Lempel-Ziv Data Compression Technique Utilizing a Dictionary Pre-Filled with Frequent Letter Combinations, Words and/or Phrases](#) (1996), [with Reynar et al.](#)

Introductory papers: I like to explain things: why computational linguistics is worth majoring in [62] and why it needs statistics [63], HMMs and forward-backward reestimation [14], the pros and cons of the Turing test [60], and how to design combinatorial optimization algorithms [61].

- [60] [Cognitive Science and the Search for Intelligence](#) (1991)
- [61] [State-of-the-Art Algorithms for Minimum Spanning Trees: A Tutorial](#) (1997)
- [62] [The Science of Language: Computational Linguistics](#) (2000)
- [63] [Introduction to the Special Section on Linguistically Apt Statistical Methods](#) (2002)

Papers by students: My students have also teamed up to publish some papers without me.

- [64] [Bilingual Parsing with Factored Estimation: Using English to Parse Korean](#) (2004), [by David Smith and Noah Smith](#)

- [65] **Context-Based Morphological Disambiguation with Random Fields** (2005),
by Noah Smith, David Smith, and Roy Tromble
- [66] **Vine Parsing and Minimum Risk Reranking for Speed and Precision** (2006),
by Markus Dreyer, David Smith, and Noah Smith