Note: In PDF and HTML versions, red hyperlinks fetch more information about a paper

Jason Eisner—Synopsis of Past Research

A central focus of my work has been **dynamic programming** for NLP.

I design **algorithms** for applying and learning **statistical models** that exploit **linguistic structure** to **improve performance on real data**.

Parsing: 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.

[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: Statistical parsing raises the question of where to get the statistical grammars. My students and I have developed several state-of-the-art approaches.

To help EM avoid poor local optima, my students and I have demonstrated the 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]. We have also 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].

When context-free grammar rules can be directly observed (in annotated Treebank data), 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, which can also be done by **generalizing latent Dirichlet allocation** [22].

Recently I proposed **strapping** [20], a technique for unsupervised model selection across many runs of bootstrapping. Strapping is remarkably accurate; it enables fully unsupervised WSD to beat lightly supervised WSD, by automatically selecting bootstrapping seeds far better than an informed human can (in fact, typically it picks the *best* seed of 200). I am now working on further machine learning innovations to reduce linguistic annotation cost, a major bottleneck in real-world applications.

[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

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 [26,27,28] that can handle imprecise ("free") translations, which are often **insufficiently parallel** to be captured by synchronous CFGs (e.g. ITGs).

A far less obvious MT-parsing connection emerges from the NP-hard problem of reordering the source-language words in an optimal way before translation. I have developed powerful **iterated local search** algorithms for such NP-hard permutation problems (as well as classical NP-hard problems like the TSP) [29]. The algorithms borrow various parsing tricks in order to explore **exponentially large local neighborhoods** in polytime.

Multilingual data is also used in some of my other recent work and that of my students [10,20,23,24,61,62,63].

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

The Dyna language (www.dyna.org): My **new programming language**, Dyna [30,31], makes it much faster and easier to build NLP 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., [11,34]) as small sets of "Horn equations." Their efficiency can be further improved by **program transformations** that derive new algorithms from old ones [32,31]. Our prototype Dyna **compiler** turns these short declarative specifications into efficient C++ classes, which instantiate my **generalized algorithms** for update propagation and automatic differentiation [31,11]. To help you understand and improve systems written in Dyna, we have also built a **visual debugger**, Dynasty, that does dynamic hypergraph layout and lets you explore huge parse forests and proof forests [33].

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 (e.g., generalizations of many lower-level implementation tricks; machine learning of effective prioritization and pruning functions).

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

Finite-state machines: I published the most general algorithms for both **training** [12,13] and **minimization** [35] of weighted finite-state machines, as well as some interesting theoretical limitations on minimization [35] and on treating multi-tape machines as infinite relational databases [37,38]. I have also developed **new finite-state approaches** in several applied contexts, including machine translation (local constraints) [29], information extraction (efficient global constraints) [39], user interfaces [37], cryptanalysis [40], and computational phonology [42,47,48]. We are designing a flexible, trainable, efficient finite-state **toolkit** to be built in Dyna [31,32,33].

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

Computational phonology and finite-state methods: I developed the **leading formalization** [41,42,44] of allowable constraints and representations in Optimality Theory (the main approach to phonology since the early 1990's [45]). This formalization, Primitive OT, permits **finite-state computation** [42] and has led to a new, **confirmed cross-linguistic prediction** [43]. I have also proposed a **modification to Optimality Theory** that further improves its computational and linguistic properties [47]. I have published key algorithms and complexity theorems on all three of the major computational problems raised by Optimality Theory: **generation** [42,47,48], **comprehension** [47,48], and **learning** [46].

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

Miscellaneous research: I have done occasional work on linguistic semantics [50], information extraction [39,51], text compression [56], collaborative filtering and online commerce [52,53,54,55], and even practical voting systems [49].

[49] Indirect STV Election: A Voting System for South Africa (1991)
[50] 'All'-less in Wonderland? Revisiting *any* (1995)
[51] Description of the University of Pennsylvania entry in the MUC-6 competition (1995)
[52] System for Generation of Object Profiles for a System for Customized Electronic Identification of Desirable Objects (1995), with Herz et al.
[53] System for Generation of User Profiles for a System for Customized Electronic Identification of Desirable Objects (1995), with Herz et al.
[54] Pseudonymous Server for System for Customized Electronic Identification of Desirable Objects (1995), with Herz et al.
[55] System for the Automatic Determination of Customized Prices and Promotions (1996), with Herz et al.
[56] 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 neat [59] and why it needs statistics [60], HMMs and forward-backward reestimation [14], the pros and cons of the Turing test [57], and how to design cool combinatorial optimization algorithms [58].

[57] Cognitive Science and the Search for Intelligence (1991)
[58] State-of-the-Art Algorithms for Minimum Spanning Trees: A Tutorial (1997)
[59] The Science of Language: Computational Linguistics (2000)
[60] 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.

[61] Bilingual Parsing with Factored Estimation: Using English to Parse Korean (2004), by David Smith and Noah Smith
[62] Context-Based Morphological Disambiguation with Random Fields (2005), by Noah Smith, David Smith, and Roy Tromble
[63] Vine Parsing and Minimum Risk Reranking for Speed and Precision (2006), by Markus Dreyer, David Smith, and Noah Smith