Probabilistic Programming and Probabilistic Databases with *Imperatively-defined Factor Graphs*

Andrew McCallum

Department of Computer Science
University of Massachusetts Amherst

Joint work with Sameer Singh, Michael Wick, Karl Schultz, Sebastian Reidel, Limin Yao, Aron Culotta.
Goal

Build models that mine actionable knowledge from unstructured text.
Rexa.info

Research • People • Connections

Andrew McCallum • Tags • Send Invites (477) • Submit • Logout

Papers • Authors • Grants

Search

Optional fields include abstract: body: title: author: venue: year: tag:
Queries may use AND, OR or (). Default is OR.

About Statistics All Tags

Sample Queries: abstract:"reinforcement learning", author:towsley, "conditional random fields".

Search and analysis on 379,011 full text papers, 7,050,439 unique paper references & 879,678 authors.

Sponsored by:

Our work in automated information extraction and co-reference is far from finished.
Please excuse the inaccuracies and missing data while we continue our work in progress.

Version 1.0 © 2008 Created by IESL, Department of Computer Science, University of Massachusetts

About Rexa • Help • Privacy Policy • Robot • Send feedback to rexa-discuss@cs.umass.edu.
1. **Table extraction using conditional random fields**
   David Pinto, Andrew McCallum, Xin Wei, W. Bruce Croft
   SIGIR, 2003
   The ability to find tables and extract information from them is a necessary component of data mining, question answering, and other information retrieval tasks. Documents often contain tables in order to communicate densely packed, multi-dimensional information. Tables do this by employing layout patterns to efficiently indicate fields and records in two-dimensional form. Their rich combination of formatting and content present difficulties for traditional language modeling techniques, however. This paper presents ... (17 citations)

2. **Learning table extraction from examples**
   A. Tengli, Yun Yang, Nianli Ma
   In Proceedings of the 20th International Conference on Computational Linguistics (COLING, 2004) (0 citations)

3. **Computational Aspects of Resilient Data Extraction from Semistructured Sources**
   Hasan Davulcu, Guizhen Yang, Michael Kifer, idhar Ramakrishnan
   PODS, 2000
   Automatic data extraction from semistructured sources such as HTML pages is rapidly growing into a problem of significant importance, spurred by the growing popularity of the so called "shopbots" that enable end users to compare prices of goods and other services at various web sites without having to manually browse and fill out forms at each one of these sites. The main problem one has to contend with when designing (5 citations)

4. **Learning Information Extraction Rules for Semi-Structured and Free Text**
   Stephen Soderland
   Machine Learning vol 34, pages 233, 1999
   A wealth of on-line text information can be made available to automatic processing by information extraction (IE) systems. Each IE application needs a separate set of rules tuned to the domain and writing style. WHISK helps to overcome this knowledge engineering bottleneck by learning text extraction rules automatically. WHISK is designed to handle text styles ranging from highly structured to free text, including text that is neither rigidly formatted nor composed (82 citations)

5. **Automatic Table Ground Truth Generation and a Background-Analysis-Based Table Structure Extraction Method**
Table extraction using conditional random fields
David Pinto, Andrew McCallum, Xin Wei, W. Bruce Croft
SIGIR, 2003 [Edit] [Email link]

Abstract:
The ability to find tables and extract information from them is a necessary component of data mining, question answering, and other information retrieval tasks. Documents often contain tables in order to communicate densely packed, multi-dimensional information. Tables do this by employing layout patterns to efficiently indicate fields and records in two-dimensional form. Their rich combination of formatting and content present difficulties for traditional language modeling techniques, however. This paper presents the use of conditional random fields (CRFs) for table extraction, and compares them with hidden Markov models (HMMs). Unlike HMMs, ...

References: (16) Sorted by date | citations | alphabetically
- Fei Sha, Fernando C N Pereira. Shallow Parsing with Conditional Random Fields. HLT-NAACL, 2003 (42 citations)
- Andrew Kachites McCallum. MALLET: a machine learning for language toolkit. 2002 (9 citations)
- David Pinto, Michael S. Brandstein, RE Coleman, W. Bruce Croft, Matthew King, Wei Li, Xin Wei. QuASM: a system for question answering using semi-structured data. JCDL, 2002 (2 citations)
- Martin J. Wainwright, Tommi Jaakkola, Alan S. Willsky. Exact MAP Estimates by (Hyper)tree Agreement. NIPS, 2002 (5 citations)
- John Daffy, Andrew McCallum, Fernando C N Pereira.

BibTex Entry: [Edit]
@inproceedings{pinto2003table,
  author = "David Pinto and Andrew McCallum and Xin Wei and W. Bruce Croft",
  title = "Table extraction using conditional random fields",
  booktitle = "SIGIR",
  pages = "235",
  year = "2003"
}

Topics:
- experimental results (20.2%), classification (13.1%), information retrieval (10.1%), speech recognition (9.1%), operations (7.1%), en automatique (6.1%), data (4%), escherichia coli (3%)

Citations: (17) Sorted by date | citations | alphabetically
- Trevor Cohn, Alvy Ray Smith, Melissa Osborne. Scaling Conditional Random Fields Using Error-Correcting Codes. Association for Computational Linguistics, pages 10-17, 2005 (2 citations)
- Charles A. Sutton, Khashayar Rohanimanesh, Andrew McCallum. Dynamic conditional random fields: factorized probabilistic models for labeling and segmenting sequence data. ICML, 2004 (8 citations)

W. Bruce Croft

Distinguished Professor
Department of Computer Science, University of Massachusetts
BRUCE CROFT, Amherst, MA, 01003-9264
Email: croftg@cs.umass.edu
URL: http://ciir.cs.umass.edu/personnel/croft.html

Publications: (1 to 40 of 233) (total 1436 citations)
Sorted by date | citations

2004
- Xiaoyong Liu, W. Bruce Croft. Cluster-based retrieval using language models. SIGIR, 2004 (0 citations)
- Andrés Corrada-Emmanuel, W. Bruce Croft. Answer models for question answering passage retrieval. SIGIR, 2004 (0 citations)
- Chirag Shah, W. Bruce Croft. Evaluating high accuracy retrieval techniques. SIGIR, 2004 (1 citation)
- Haizheng Zhang, W. Bruce Croft, Brian N. Levine, Victor R. Lesser. A Multi-Agent Approach for Peer-to-Peer Based Information Retrieval System. AAMAS, 2004 (1 citation)
- Donald Metzler, Victor Lavrenko, W. Bruce Croft. Formal multiple-bernoulli models for language modeling. SIGIR, 2004 (0 citations)
- Stephen Cronen-Townsend, Yu Zhou, W. Bruce Croft. A framework for selective query expansion. CIKM, 2004 (0 citations)

2003
- W. Bruce Croft. Language Models for Information Retrieval. CIKM, 2003 (0 citations)
CISE Research Infrastructure: Infrastructure to Support Research on Networked Multimedia Information Systems

James F. Kurose, John A. Stankovic, Donald F. Towsley, Krithi Ramamritham, J. Eliot B Moss, W. Richards Adrion, W. Bruce Croft, Kathryn McKinley
NSF Grant EIA-9502639, August 1, 1995 - December 29, 1999

Abstract:

This award provides support to equip a networked, experimental testbed to enable research in the development of the operating system, I/O, networking, object management, and information retrieval components of future networked multimedia information systems. The testbed will consist of two shared-memory multiprocessor facilities attached to several parallel mass storage I/O devices and a high-speed ATM network. The research team will be developing several key hardware and software technologies needed to support future networked, multimedia information systems. Specific research areas include operating systems, I/O, networking, object management and information retrieval.

Papers: (17) Sorted by date | citations | alphabetically

- Emery D. Berger, Benjamin G. Zorn, Kathryn S. McKinley. Composing High-Performance Memory Allocators. PLDI, 2001 (7 citations)
- Sally Floyd, Mark Handley, Jitendra Padhye, Jörg Widmer. Equation-based congestion control for unicast applications. SIGCOMM, 2000 (229 citations)
- Sally Floyd, Mark Handley, Jitendra Padhye. Equation-Based Congestion Control for Unicast Applications \(\lambda\) Lambda. 2000 (7 citations)
- Supratik Bhattacharyya, Don Towsley, James F. Kurose. Design and Analysis of Loss Indication Filters for Multicast Congestion Control. CMPSCI Technical Report TR 99-46, Department of Computer Science University of Massachusetts Amherst, 2000 (0 citations)
- Brendon Cahoon, Kathryn S. McKinley. Tolerating Latency by Prefetching Java Objects. To appear: Workshop on Hardware Support for Objects and Microarchitectures for Java, 1999 (3 citations)
- Jitendra Padhye, James F. Kurose, Donald F. Towsley, Rajeev Koodli. A TCP-Friendly Rate Adjustment Protocol for Continuous Media Flows over Best Effort Networks CMPSCI
Entities and Relations

Research Paper

Cites
Entities and Relations

Grant - Research Paper
Venue - Research Paper
University - Person
Groups - Person
Expertise - Person

Cites
Extracted Database

- 8 million research papers
- 2 million authors
- 400k grants, 90k institutions, 10k venues
Applying probabilistic modeling
Applying probabilistic modeling to large data.

Information extraction  bio/medical informatics  computer vision  scientific data modeling  ...

Computational Statistics

- scalability
- data structures
- algorithms
- parallelism

Rich model structure
- spatio-temporal
- hierarchical
- relational
- infinite

Implementing the new model is a significant task.

Software engineering
Bayesian Network
Directed, Generative Graphical Models

\[ p(a, b, c, d, e) = p(a) \ p(b|a) \ p(c|a) \ p(d|a, c) \ p(e|b, c) \]
Markov Random Field

Undirected Graphical Models, aka, Markov Network

\[ p(a, b, c, d, e) = \frac{1}{Z} \phi(a, c, d) \phi(a, b) \phi(b, e) \phi(d, e) \]
Markov Random Field

Undirected Graphical Models, aka, Markov Network

\[ p(a, b, c, d, e) = \frac{1}{Z} \phi(a, c, d) \phi(a, b) \phi(b, e) \phi(d, e) \]

\[ p(a, b, c, d, e) = \frac{1}{Z} \phi(a, c) \phi(a, d) \phi(c, d) \phi(a, b) \phi(b, e) \phi(d, e) \]
Factor Graph

Can represent both directed and undirected graphical models

\[
p(a, b, c, d, e) = \frac{1}{Z} \phi(a, c, d) \phi(a, b) \phi(b, e) \phi(d, e)
\]

\[
p(a, b, c, d, e) = \frac{1}{Z} \phi(a, c)\phi(a, d)\phi(c, d) \phi(a, b) \phi(b, e) \phi(d, e)
\]
Factor Graph

Can represent both directed and undirected graphical models

\[ p(a, b, c, d, e) = \frac{1}{Z} \phi(a, c, d) \phi(a, b) \phi(b, e) \phi(d, e) \]

\[ p(a, b, c, d, e) = \frac{1}{Z} \phi(a, c)\phi(a, d)\phi(c, d) \phi(a, b) \phi(b, e) \phi(d, e) \]
Conditional Random Field (CRF)

Undirected graphical model, conditioned on some data variables

\[ p(y|x) = \frac{1}{Z_x} \prod_{f} \phi(x \in f, y \in f) \]

[Lafferty, McCallum, Pereira 2001]
Conditional Random Field (CRF)

Undirected graphical model, conditioned on some data variables

\[ p(y|x) = \frac{1}{Z_x} \prod_{f} \phi(x \in f, y \in f) \]

+ Tremendous freedom to use arbitrary features of input.
+ Predict multiple dependent variables ("structured output")

[Lafferty, McCallum, Pereira 2001]
Relational Graphical Model

*Relational* = repeated structure of data & factors
Relational Graphical Model

*Relational* = repeated structure of data & factors
Information Extraction with Linear-chain CRFs

Graphical model

Finite state model

State-of-the-art predictive accuracy on many tasks.
Outline

- Motivate software engineering for statistics
- Graphical models for Extraction & Integration
  - Extraction (linear-chain CRFs)
  - Information Integration (really hairy CRFs, MCMC, SampleRank)
- Probabilistic Programming: FACTORIE
- Example
- Relation Extraction (cross-document, w/out labeled data)
- Probabilistic Programming inside a DB
- Ongoing Work
• Motivate software engineering for statistics
• Graphical models for Extraction & Integration
  - Extraction (linear-chain CRFs)
  - Information Integration (really hairy CRFs, MCMC, SampleRank)
• Probabilistic Programming: FACTORIE
• Example
• Relation Extraction (cross-document, w/out labeled data)
• Probabilistic Programming inside a DB
• Ongoing Work
## Information Integration

### Database A (Schema A)

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>J.</td>
<td>Smith</td>
<td>222-444-1337</td>
</tr>
<tr>
<td>J.</td>
<td>Smith</td>
<td>444 1337</td>
</tr>
<tr>
<td>John</td>
<td>Smith</td>
<td>(1) 4321115555</td>
</tr>
</tbody>
</table>

### Database B (Schema B)

<table>
<thead>
<tr>
<th>Name</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Smith</td>
<td>U.S. 222-444-1337</td>
</tr>
<tr>
<td>John D. Smith</td>
<td>444 1337</td>
</tr>
<tr>
<td>J Smiht</td>
<td>432-111-5555</td>
</tr>
</tbody>
</table>

### Schema Matching

<table>
<thead>
<tr>
<th>Schema A</th>
<th>Schema B</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
<td>Name</td>
</tr>
<tr>
<td>Last Name</td>
<td>Phone</td>
</tr>
<tr>
<td>Contact</td>
<td></td>
</tr>
</tbody>
</table>

### Coreference

**John #1**
- J. Smith
- J. Smith
- John Smith
- John D. Smith

**John #2**
- John Smith
- J Smiht

### Canonicalization: Normalized DB

<table>
<thead>
<tr>
<th>Entity#</th>
<th>Name</th>
<th>Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>523</td>
<td>John Smith</td>
<td>222-444-1337</td>
</tr>
<tr>
<td>524</td>
<td>John D. Smith</td>
<td>432-111-5555</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
• $x_1$ is a set of mentions \{J. Smith, John, John Smith\}
• $x_2$ is a set of mentions \{Amanda, A. Jones\}
• $f_{12}$ is a factor between $x_1/x_2$
• $y_{12}$ is a binary variable indicating a match (no)
• $f_1$ is a factor over cluster $x_1$
• $y_1$ is a binary variable indicating match (yes)
• Entity/attribute factors omitted for clarity
Schema Matching

\[ P(Y \mid X) = \frac{1}{Z_X} \prod_{y_i \in Y} \psi_n(y_i, x_i) \prod_{y_i, y_j \in Y} \psi_m(y_i, x_{ij}) \]

\[ \psi(y_i, x_i) = \exp \left( \sum_k \lambda_k f_k(y_i, x_i) \right) \]

- \( x_6 \) is a set of attributes \{phone, contact, telephone\}
- \( x_7 \) is a set of attributes \{last name, last name\}
- \( f_{67} \) is a factor between \( x_6/x_7 \)
- \( y_{67} \) is a binary variable indicating a match (no)
- \( f_7 \) is a factor over cluster \( x_7 \)
- \( y_7 \) is a binary variable indicating match (yes)
Schema Matching

\[ P(Y \mid X) = \frac{1}{Z_X} \prod_{y_i \in Y} \psi_n(y_i, x_i) \prod_{y_i, y_j \in Y} \psi_b(y_{ij}, x_{ij}) \]

\[ \psi(y_i, x_i) = \exp \left( \sum_k \lambda_k f_k(y_i, x_i) \right) \]

Coreference and Canonicalization
Really Hairy!

How to do

• parameter estimation
• inference
Parameter Estimation in Large State Spaces

- Most methods require calculating gradient of log-likelihood, \( P(y_1, y_2, y_3, \ldots | x_1, x_2, x_3, \ldots) \)
- ...which in turn requires “expectations of marginals,” \( P(y_1 | x_1, x_2, x_3, \ldots) \)
- But, getting marginal distributions can be difficult.

- Alternative: Perceptron. Approximate gradient from difference between true output and model’s predicted best output.
- But, even finding model’s predicted best output is expensive.

- We propose: “Sample Rank” [Culotta, Wick, Hall, McCallum, HLT 2007]
  Learn to rank intermediate solutions:
  \( P(y_1=1, y_2=0, y_3=1, \ldots | \ldots) > P(y_1=0, y_2=0, y_3=1, \ldots | \ldots) \)
Metropolis-Hastings

Given factor graph with target variables \( y \) and observed \( x \)

\[
P(y|x) = \frac{1}{Z_x} \prod_{y^i \in \mathcal{F}} \psi(x, y^i)
\]

\( \mathcal{F} \) feasible region defined by deterministic constraints
e.g. clustering, parse-tree projectivity.

\( q \) proposal distribution \( q(y'|y) : \mathcal{F} \times \mathcal{F} \to [0, 1] \)

1. Begin with some initial configuration \( y_0 \in \mathcal{F} \)
2. For \( i = 1, 2, 3, \ldots \) draw a local modification \( y' \in \mathcal{F} \) from \( q \)
3. Probabilistically accept mod as Bernoulli draw with param \( \alpha \)

\[
\alpha = \min \left( 1, \frac{p(y')}{{p(y)}} \frac{q(y|y')}{{q(y'|y)}} \right)
\]

Can do MAP inference with decreasing temperature on ratio of \( p(y) \)'s
1. Partition function cancels

\[
\frac{p(y')}{p(y)} = \frac{p(Y = y'|x; \theta)}{p(Y = y|x; \theta)} = \frac{1}{Z_x} \prod_{y^i \in y'} \psi(x, y^i) \frac{1}{Z_x} \prod_{y \in y} \psi(x, y^i) = \frac{\prod_{y^i \in y'} \psi(x, y^i)}{\prod_{y \in y} \psi(x, y^i)}
\]

2. Unchanged factors cancel

\[
\frac{\prod_{y^i \in \delta_y} \psi(x, y'^i)}{\prod_{y^i \in y/\delta_y} \psi(x, y^i)} = \frac{\prod_{y^i \in \delta_y} \psi(x, y'^i)}{\prod_{y^i \in y/\delta_y} \psi(x, y^i)} \frac{\prod_{y^i \in \delta_y} \psi(x, y^i)}{\prod_{y^i \in y/\delta_y} \psi(x, y^i)} = \frac{\prod_{y^i \in \delta_y} \psi(x, y'^i)}{\prod_{y^i \in \delta_y} \psi(x, y^i)}
\]
Like Perceptron:
Proof of convergence under Marginal Separability.

More constrained than Maximum Likelihood:
Parameters must correctly rank *incorrect* solutions!

Very fast to train.
Comparison to Contrastive Divergence

Contrastive Divergence, $n=2$  
[Hinton 2002]

Persistent Contrastive Divergence  
[Tieleman 2008]

Sample Rank
Comparison to LASO

LASO scores “actions” (transitions).

Sample Rank scores possible worlds.

No concern about generation ordering of output vars. Defines standard factor graph score on possible world. Can get marginal probability distributions.

Daumé, Langford, Marcu ’05
SampleRank on Coreference

- ACE 2004
- All nouns. 28,122 mentions, 14,047 entities
  e.g. he, the President, Clinton, Mrs. Clinton, Washington

<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Method</th>
<th>B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005</td>
<td>Ng</td>
<td></td>
<td>69.5%</td>
</tr>
<tr>
<td>2007</td>
<td>Culotta, Wick, Hall, McCallum</td>
<td></td>
<td>79.3%</td>
</tr>
<tr>
<td>2008</td>
<td>Bengston, Roth</td>
<td></td>
<td>80.8%</td>
</tr>
<tr>
<td>2009</td>
<td>Wick, McCallum</td>
<td>MCMC+SampleRank</td>
<td>81.5%</td>
</tr>
</tbody>
</table>
\[ P(Y \mid X) = \frac{1}{Z_X} \prod_{y_i \in Y} \psi_w(y_i, x_i) \prod_{y_i, y_j \in Y} \psi_b(y_{ij}, x_{ij}) \]

\[ \psi(y_i, x_i) = \exp \left( \sum_k \lambda_k f_k(y_i, x_i) \right) \]
Dataset

- Faculty and alumni listings from university websites, plus an IE system
- 9 different database schemas
- \(~1400\) mentions, 294 coreferent

Example schemas:

<table>
<thead>
<tr>
<th>DEX IE</th>
<th>Northwestern Fac</th>
<th>UPenn Fac</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
<td>Name</td>
<td>Name</td>
</tr>
<tr>
<td>Middle Name</td>
<td>Title</td>
<td>First Name</td>
</tr>
<tr>
<td>Last Name</td>
<td>PhD Alma Mater</td>
<td>Last Name</td>
</tr>
<tr>
<td>Title</td>
<td>Research Interests</td>
<td>Job+Department</td>
</tr>
<tr>
<td>Department</td>
<td></td>
<td>Office Address</td>
</tr>
<tr>
<td>Company Name</td>
<td></td>
<td>E-mail</td>
</tr>
<tr>
<td>Home Phone</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Office Phone</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fax Number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E-mail</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Coreference Results

<table>
<thead>
<tr>
<th></th>
<th>Pair</th>
<th></th>
<th>MUC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1</td>
<td>Prec</td>
<td>Recall</td>
</tr>
<tr>
<td>No Canon</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISO</td>
<td>72.7</td>
<td>88.9</td>
<td>61.5</td>
</tr>
<tr>
<td>CASC</td>
<td>64.0</td>
<td>66.7</td>
<td>61.5</td>
</tr>
<tr>
<td>JOINT</td>
<td>76.5</td>
<td>89.7</td>
<td>66.7</td>
</tr>
<tr>
<td>Canon</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISO</td>
<td>78.3</td>
<td>90.0</td>
<td>69.2</td>
</tr>
<tr>
<td>CASC</td>
<td>65.8</td>
<td>67.6</td>
<td>64.1</td>
</tr>
<tr>
<td>JOINT</td>
<td>81.7</td>
<td>90.6</td>
<td>74.4</td>
</tr>
</tbody>
</table>

ISO = isolated  CASC = cascade  JOINT = joint inference

~15% error reduction from joint model
## Schema Matching Results

<table>
<thead>
<tr>
<th>Pair</th>
<th>ISO</th>
<th>CASC</th>
<th>JOINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Canon</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td>50.9</td>
<td>50.9</td>
<td>68.9</td>
</tr>
<tr>
<td>Prec</td>
<td>40.9</td>
<td>40.9</td>
<td>100</td>
</tr>
<tr>
<td>Recall</td>
<td>67.5</td>
<td>67.5</td>
<td>52.5</td>
</tr>
<tr>
<td>MUC</td>
<td>69.2</td>
<td>69.2</td>
<td>69.6</td>
</tr>
<tr>
<td>F1</td>
<td>69.2</td>
<td>69.2</td>
<td>75.0</td>
</tr>
<tr>
<td>Prec</td>
<td>81.8</td>
<td>81.8</td>
<td>100</td>
</tr>
<tr>
<td>Recall</td>
<td>60.0</td>
<td>60.0</td>
<td>60.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Canon</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
</tr>
<tr>
<td>Prec</td>
</tr>
<tr>
<td>Recall</td>
</tr>
<tr>
<td>MUC</td>
</tr>
<tr>
<td>F1</td>
</tr>
<tr>
<td>Prec</td>
</tr>
</tbody>
</table>

ISO = isolated  CASC = cascade  JOINT = joint inference

~40% error reduction from joint model
Schema Matching

\[
P(Y \mid X) = \frac{1}{Z_x} \prod_{y \in Y} \psi_n(y_i, x_i) \prod_{y_i, y \in Y} \psi_m(y_{ij}, x_{ij})
\]

\[
\psi(y_i, x_i) = \exp \left( \sum_k \lambda_{fi}(y_i, x_i) \right)
\]

Really Hairy!
How to do

✔ • parameter estimation
✔ • inference

Coreference and Canonicalization
Really Hairy!

How to do

✔ • parameter estimation
✔ • inference
✔ • software engineering

Coreference and Canonicalization

$$P(Y | X) = \frac{1}{Z_X} \prod_{y \in Y} \psi_n(y_i, x_i) \prod_{y, \gamma \in Y} \psi_n(y_i, x_{ij})$$

$$\psi(y_i, x_i) = \exp \left( \sum_k \lambda_i f_i(y_i, x_i) \right)$$
Outline

• Motivate software engineering for statistics

• Graphical models for Extraction & Integration
  - Extraction (linear-chain CRFs)
  - Information Integration (really hairy CRFs, MCMC, SampleRank)

• Probabilistic Programming: FACTORIE

• Example

• Relation Extraction (cross-document, w/out labeled data)

• Probabilistic Programming inside a DB

• Ongoing Work
Probabilistic Programming Languages

• Make it easy to specify rich, complex models, using the full power of programming languages
  - data structures
  - control mechanisms
  - abstraction

• Inference implementation comes for free

Provides language to easily create new models
Small Sampling of Probabilistic Programming Languages

• Explicit Directed Graph
  - BUGS

• Functional
  - IBAL, Church

• Object Oriented
  - Figaro, Infer.NET

• Logic-based
  - Markov logic, BLOG, PRISM
Declarative Model Specification

• One of biggest advances in Artificial Intelligence community

• Gone too far?
  Much domain knowledge is also procedural.

• Logic + Probability $\rightarrow$ Imperative + Probability
  - Rising interest: Church, Infer.NET,...
Imperative tools for creating a Declarative Model Specification

- One of biggest advances in Artificial Intelligence community

- Gone too far?
  Much domain knowledge is also procedural.

- Logic + Probability $\rightarrow$ Imperative + Probability
  - Rising interest: Church, Infer.NET,...
Imperative tools for creating a Declarative Model Specification

Our approach:

- Preserve the *declarative* statistical semantics of factor graphs
- Provide *imperative* hooks to define structure, parameterization, inference, estimation.

“Imperatively-Defined Factor Graphs”

[McCallum, Rohanemanesh, Wick, Schultz, Singh, NIPS, 2008]
Our Design Goals

- Represent factor graphs
  - emphasis on conditional random fields
- Scalability
  - input data, output configuration, factors, tree-width
  - observed data that cannot fit in memory
  - super-exponential number of factors
- Leverage object-oriented benefits
  - Modularity, encapsulation, inheritance,...
- Integrate declarative & procedural knowledge
  - natural, easy-to-use
  - upcoming slides:
  2 examples of injecting imperativ-ism into factor graphs
• “Factor Graphs, Imperative, Extensible”
• Implemented as a library in Scala [Martin Odersky]
  - object oriented & functional
  - type inference
  - runs in JVM (complete interoperation with Java)
  - fast, JIT compiled, but also cmd-line interpreter
• Library, not new “little language”
  - integrate data pre-processing & eval. w/ model spec
  - leverage OO-design: modularity, encapsulation, inheritance
• Scalable
  - billions of variables, super-exp #factors, DB back-end
  - fast parameter estimation through SampleRank [2009]
Stages of FACTORIE programming

1. Define “templates for data” (i.e. classes)
   - Use data structures just like in deterministic programming.
   - Only special requirement: “undo” capability for changes.
   - (Variable holds single possible value, not a distribution.)

2. Define “templates for factors”
   - Distinct from above data representation; makes it easy to modify model scoring independently.
   - Leverage data’s natural relations to define factors’ relations.

3. Select inference (MCMC, variational)
   - Optionally, define MCMC proposal functions that leverage domain knowledge.

4. Read the data, creating variables.
   Then inference / parameter estimation is often a one-liner!
Outline

• Motivate software engineering for statistics
• Graphical models for Extraction & Integration
  - Extraction (linear-chain CRFs)
  - Information Integration (really hairy CRFs, MCMC, SampleRank)
• Probabilistic Programming: FACTORIE
• Example
• Relation Extraction (cross-document, w/out labeled data)
• Probabilistic Programming inside a DB
• Ongoing Work
Scala

• New variable
  var myHometown : String

• New constant
  val myName = "Andrew"

• New method
  def climb(increment: double) = myAltitude += increment

• New class
  class Skier extends Person

• New trait (like Java interface with implementations)
  trait FirstAid { def applyBandage = ... }

• New class with trait
  class BackcountrySkier extends Skier with FirstAid

• Generics in square brackets
  new ArrayList[Skier]
Example: Linear-Chain CRF for Segmentation

Labels

Words

Bill  loves  skiing  Tom  loves  snowshoeing
Example: Linear-Chain CRF for Segmentation

class Label(isBeg:boolean) extends BooleanVariable(isBeg)

class Token(word:String) extends CategoricalVariable(word)
Example: Linear-Chain CRF for Segmentation

class Label(isBeg:boolean) extends BooleanVariable(isBeg) with VarInSeq

class Token(word:String) extends CategoricalVariable(word) with VarInSeq

label.prev label.next
Example: Linear-Chain CRF for Segmentation

class Label(isBeg:boolean) extends BooleanVariable(isBeg) with VarInSeq

class Token(word:String) extends CategoricalVariable(word) with VarInSeq
Example: Linear-Chain CRF for Segmentation

class Label(isBeg: boolean) extends BooleanVariable(isBeg) with VarInSeq {
    val token : Token
}
class Token(word: String) extends CategoricalVariable(word) with VarInSeq {
    val label : Label
}

Avoid representing relations by indices.
Do it directly with member pointers... arbitrary data structure.
Example: Linear-Chain CRF for Segmentation

class Label(isBeg:boolean) extends BooleanVariable(isBeg) with VarInSeq {
  val token : Token
}
class Token(word:String) extends CategoricalVariable(word) with VarInSeq {
  val label : Label
  def longerThanSix = word.length > 6
}

Bill loves skiing Tom loves snowshoeing

Labels

Words

T  F  F  T  F  F
Example: Linear-Chain CRF for Segmentation

class Label(isBeg:boolean) extends BooleanVariable(isBeg) with VarInSeq {
  val token : Token
}
class Token(word:String) extends CategoricalVariable(word) with VarInSeq {
  val label : Label
  def longerThanSix = word.length > 6
}

val model = new Model(
  new TemplateWithStatistics1[Label]
)
Example: Linear-Chain CRF for Segmentation

class Label(isBeg:boolean) extends BooleanVariable(isBeg) with VarInSeq {
  val token : Token
}
class Token(word:String) extends CategoricalVariable(word) with VarInSeq {
  val label : Label
  def longerThanSix = word.length > 6
}

val model = new Model(
  new TemplateWithStatistics1[Label],
  new TemplateWithStatistics2[Label,Token]
)
Example: Linear-Chain CRF for Segmentation

class Label(isBeg:Boolean) extends BooleanVariable(isBeg) with VarInSeq {
    val token : Token
}
class Token(word:String) extends CategoricalVariable(word) with VarInSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

val model = new Model(
    new TemplateWithStatistics1[Label],
    new TemplateWithStatistics2[Label,Token],
    new TemplateWithStatistics2[Label,Label]
)

<table>
<thead>
<tr>
<th>Labels</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>Bill</td>
</tr>
<tr>
<td>F</td>
<td>loves</td>
</tr>
<tr>
<td>F</td>
<td>skiing</td>
</tr>
<tr>
<td>T</td>
<td>Tom</td>
</tr>
<tr>
<td>F</td>
<td>loves</td>
</tr>
<tr>
<td>F</td>
<td>snowshoeing</td>
</tr>
</tbody>
</table>
Key Operation: Scoring a Proposal

• Acceptance probability ~ ratio of model scores. Scores of factors that didn’t change cancel.

• To efficiently score:
  – Proposal method runs.
Key Operation: Scoring a Proposal

• Acceptance probability \( \sim \) ratio of model scores. Scores of factors that didn’t change cancel.

• To efficiently score:
  – Proposal method runs.
  – Automatically build a list of variables that changed.
Key Operation: Scoring a Proposal

- Acceptance probability ~ ratio of model scores. Scores of factors that didn’t change cancel.
- To efficiently score:
  - Proposal method runs.
  - Automatically build a list of variables that changed.
  - Find factors that touch changed variables
Key Operation: Scoring a Proposal

- Acceptance probability ~ ratio of model scores. Scores of factors that didn’t change cancel.
- To efficiently score:
  - Proposal method runs.
  - Automatically build a list of variables that changed.
  - Find factors that touch changed variables
  - Find other (unchanged) variables needed to calculate those factors’ scores
Key Operation: Scoring a Proposal

- Acceptance probability ~ ratio of model scores. Scores of factors that didn’t change cancel.
- To efficiently score:
  - Proposal method runs.
  - Automatically build a list of variables that changed.
  - Find factors that touch changed variables
  - Find other (unchanged) variables needed to calculate those factors’ scores
Key Operation: Scoring a Proposal

• Acceptance probability ~ ratio of model scores. Scores of factors that didn’t change cancel.

• To efficiently score:
  – Proposal method runs.
  – Automatically build a list of variables that changed.
  – Find factors that touch changed variables
  – Find other (unchanged) variables needed to calculate those factors’ scores

• How to find factors from variables & vice versa?
  – In BLOG, rich, highly-indexed data structure stores mapping variables ←→ factors
  – But complex to maintain as structure changes
  – Factors consume memory
Imperativ-ism #1: Model Structure

• Maintain no map structure between factors and variables

• Finding factors is easy. Usually # templates < 50.
  – Given changed variable, query each template

• **Primitive operation:**
  Given factor template and one changed variable, find other variables

• In factor Template object, define *imperative methods* that do this.
  – `unroll1(v1)` returns `(v1,v2,v3)`
  – `unroll2(v2)` returns `(v1,v2,v3)`
  – `unroll3(v3)` returns `(v1,v2,v3)`
  – i.e., use Turing-complete language to determine structure on the fly.

• Other nice attributes
  – Easy to do value-conditioned structure. Case Factor Diagrams, etc.
  – Not only avoid super-exp, don’t even allocate all factors for current config.
  – FACTORIE provides several simpler mechanisms that build on this primitive.
Example: Linear-Chain CRF for Segmentation

```scala
class Label(isBeg: boolean) extends BooleanVariable(isBeg) with VarInSeq {
  val token : Token
}

class Token(word: String) extends CategoricalVariable(word) with VarInSeq {
  val label : Label
  def longerThanSix = word.length > 6
}

val model = new Model(
  new TemplateWithStatistics1[Label],
  new TemplateWithStatistics2[Label, Token],
  new TemplateWithStatistics2[Label, Label]
)
```

Example: Linear-Chain CRF for Segmentation

```scala
class Label(isBeg: boolean) extends BooleanVariable(isBeg) with VarInSeq {
    val token : Token
}
class Token(word: String) extends CategoricalVariable(word) with VarInSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

val model = new Model(
    new TemplateWithStatistics1[Label],
    new TemplateWithStatistics2[Label, Token] {
    },
    new TemplateWithStatistics2[Label, Label]
)
```
Example: Linear-Chain CRF for Segmentation

class Label(isBeg:boolean) extends BooleanVariable(isBeg) with VarInSeq {
    val token : Token
}

class Token(word:String) extends CategoricalVariable(word) with VarInSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

val model = new Model(
    new TemplateWithStatistics1[Label],
    new TemplateWithStatistics2[Label,Token] {
        def unroll1(label:Label) = Factor(label, label.token)
    },
    new TemplateWithStatistics2[Label,Label]
)
Example: Linear-Chain CRF for Segmentation

class Label(isBeg: boolean) extends BooleanVariable(isBeg) with VarInSeq {
    val token : Token
}
class Token(word: String) extends CategoricalVariable(word) with VarInSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

val model = new Model(
    new TemplateWithStatistics1[Label],
    new TemplateWithStatistics2[Label, Token] {
        def unroll1(label: Label) = Factor(label, label.token)
        def unroll2(token: Token) = throw new Error // Tokens shouldn’t change
    },
    new TemplateWithStatistics2[Label, Label]
)
Example: Linear-Chain CRF for Segmentation

class Label(isBeg:boolean) extends BooleanVariable(isBeg) with VarInSeq {
    val token : Token
}
class Token(word:String) extends CategoricalVariable(word) with VarInSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

val model = new Model(
    new TemplateWithStatistics1[Label],
    new TemplateWithStatistics2[Label,Token] {
        def unroll1(label:Label) = Factor(label, label.token)
        def unroll2(token:Token) = throw new Error // Tokens shouldn't change
    },
    new TemplateWithStatistics2[Label,Label] {
        def unroll1(label:Label) = Factor(label, label.next)
    }
)
Example: Linear-Chain CRF for Segmentation

class Label(isBeg: boolean) extends BooleanVariable(isBeg) with VarInSeq {
    val token : Token
}
class Token(word: String) extends CategoricalVariable(word) with VarInSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

val model = new Model(
    new TemplateWithStatistics1[Label],
    new TemplateWithStatistics2[Label, Token] {
        def unroll1(label: Label) = Factor(label, label.token)
        def unroll2(token: Token) = throw new Error // Tokens shouldn't change
    },
    new TemplateWithStatistics2[Label, Label] {
        def unroll1(label: Label) = Factor(label, label.next)
        def unroll2(label: Label) = Factor(label.prev, label)
    }
    )

![Diagram of Labels and Words]
Imperativ-ism #2: Neighbor-Sufficient Map

• “Neighbor Variables” of a factor
  – Values of variables touching the factor
• “Sufficient Statistics” of a factor
  – Vector, dot product with weights of log-linear factor $\rightarrow$ factor’s score

• Usually confounded. Separate them w/ user-defined function!
• Skip-chain NER. Instead of 5x5 parameters, just 2.
  $(\text{label}_1, \text{label}_2) \rightarrow \text{label}_1 == \text{label}_2$

[Sutton & McCallum 2006]
Example: Skip-Chain CRF for Segmentation

```scala
class Label(isBeg:boolean) extends BooleanVariable(isBeg) with VarInSeq {
  val token : Token
}
class Token(word:String) extends CategoricalVariable(word) with VarInSeq {
  val label : Label
  def longerThanSix = word.length > 6
}

val model = new Model(
  new TemplateWithStatistics1[Label],
  new TemplateWithStatistics2[Label,Token] {
    def unroll1(label:Label) = Factor(label, label.token)
    def unroll2(token:Token) = throw new Error // Tokens shouldn’t change
  },
  new TemplateWithStatistics2[Label,Label] {
    def unroll1(label:Label) = Factor(label, label.next)
    def unroll2(label:Label) = Factor(label.prev, label)
  }
)
```
Example: Skip-Chain CRF for Segmentation

class Label(isBeg:boolean) extends BooleanVariable(isBeg) with VarInSeq {
    val token : Token
}
class Token(word:String) extends CategoricalVariable(word) with VarInSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

val model = new Model(
    new TemplateWithStatistics1[Label],
    new TemplateWithStatistics2[Label,Token] {
        def unroll1(label:Label) = Factor(label, label.token)
        def unroll2(token:Token) = throw new Error // Tokens shouldn't change
    },
    new TemplateWithStatistics2[Label,Label] {
        def unroll1(label:Label) = Factor(label, label.next)
        def unroll2(label:Label) = Factor(label.prev, label)
    },
    new Template2[Label,Label] with Statistics1[BooleanVariable]
)
Example: Skip-Chain CRF for Segmentation

class Label(isBeg:boolean) extends BooleanVariable(isBeg) with VarInSeq {
    val token : Token
}
class Token(word:String) extends CategoricalVariable(word) with VarInSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

val model = new Model(
    new TemplateWithStatistics1[Label],
    new TemplateWithStatistics2[Label,Token] {
        def unroll1(label:Label) = Factor(label, label.token)
        def unroll2(token:Token) = throw new Error // Tokens shouldn’t change
    },
    new TemplateWithStatistics2[Label,Label] {
        def unroll1(label:Label) = Factor(label, label.next)
        def unroll2(label:Label) = Factor(label.prev, label)
    },
    new Template2[Label,Label] with Statistics1[BooleanVariable]{
        def unroll1(label:Label) = for (other <- label.seq;
            if (label.token == other.token)) yield Factor(label,other)
    }
)
Example: Skip-Chain CRF for Segmentation

class Label(isBeg: boolean) extends BooleanVariable(isBeg) with VarInSeq {
  val token : Token
}
class Token(word: String) extends CategoricalVariable(word) with VarInSeq {
  val label : Label
  def longerThanSix = word.length > 6
}

val model = new Model(
  new TemplateWithStatistics1[Label],
  new TemplateWithStatistics2[Label, Token] {
    def unroll1(label: Label) = Factor(label, label.token)
    def unroll2(token: Token) = throw new Error // Tokens shouldn’t change
  },
  new TemplateWithStatistics2[Label, Label] {
    def unroll1(label: Label) = Factor(label, label.next)
    def unroll2(label: Label) = Factor(label.prev, label)
  },
  new Template2[Label, Label] with Statistics1[BooleanVariable]{
    def unroll1(label: Label) = for (other <- label.seq;
                                        if (label.token == other.token)) yield Factor(label, other)
    def statistics(label1: Label, label2: Label) = Stat(label1 == label2)
  }
)
Example: Skip-Chain CRF for Segmentation

class Label(isBeg:boolean) extends BooleanVariable(isBeg) with VarInSeq {
    val token : Token
}
class Token(word:String) extends CategoricalVariable(word) with VarInSeq {
    val label : Label
    def longerThanSix = word.length > 6
}

val model = new Model(
    new TemplateWithStatistics1[Label],
    new TemplateWithStatistics2[Label,Token] {
        def unroll1(label:Label) = Factor(label, label.token)
        def unroll2(token:Token) = throw new Error // Tokens shouldn’t change
    },
    new TemplateWithStatistics2[Label,Label] {
        def unroll1(label:Label) = Factor(label, label.next)
        def unroll2(label:Label) = Factor(label.prev, label)
    },
        def unroll1(label:Label) = for (other <- label.seq;
            if (label.token == other.token)) yield Factor(label,other)
        def statistics(label1:Label, label2:Label) = Stat(label1 == label2)
    })

val labels:Collection[Label] = readData()
val inferencer = new GibbsSampler(model)
for (i <- 1 to numIterations) inferencer.process(labels)
Example Run

CoNLL 2003 NER
Example: Dependency Parsing

class Word(str: String) extends CategoricalVariable(word)
class Node(word: Word, parent: Node) extends RefVariable(parent)

  def statistics(n: Node) = Stat(n.word, n.parent.word)
}

  def statistics(n: Node) = Stat(n.word, closestVerb(n).word)
  def closestVerb(n: Node) = if (isVerb(n.word)) n else closestVerb(n.parent)
  def unroll1(n: Node) = n.selfAndDescendants
}
Example: Alternative Template Spec

Instead of previous:

```scala
val model = new Model(
    new TemplateWithStatistics1[Label],
    new TemplateWithStatistics2[Label,Token] {
        def unroll1(label:Label) = Factor(label, label.token)
        def unroll2(token:Token) = throw new Error
    },
    new TemplateWithStatistics2[Label,Label] {
        def unroll1(label:Label) = Factor(label, label.next)
        def unroll2(label:Label) = Factor(label.prev, label)
    },
)
```

Higher-level “Entity-Relationship” Specification:

```scala
val model = new Model(
    Foreach[Label] { label => Score(label) },
    Foreach[Label] { label => Score(label, label.token) },
    Foreach[Label] { label => Score(label.prev, label) }
)
Example: First-order Logic Templates

```scala
val model = new Model {
  Forany[Person] { p => p.cancer } * 0.1,
  Forany[Person] { p => p.smokes ==> p.cancer } * 2.0
  Forany[Person] { p => p.friends.smokes <=> p.smokes } * 1.5
}
```
Example: First-order Logic Templates

val model = new Model (  
    Forany[Person] { p => p.cancer } * 0.1,  
    Forany[Person] { p => p.smokes ==> p.cancer } * 2.0  
    Forany[Person] { p => p.friends.smokes <=> p.smokes } * 1.5  
  )
Example: Latent Dirichlet Allocation

class $Z(p:Proportions, value:Int)$
    extends MixtureChoice(p, value)
class Word(ps:FiniteMixture[Proportions], $z$:MixtureChoiceVariable, value:String)
    extends CategoricalMixture[String](ps, $z$, value)
class Document(val file:String)
    extends ArrayBuffer[Word] { var theta:DirichletMultinomial = null }
Example: Latent Dirichlet Allocation

class Z(p:Proportions, value:Int)
   extends MixtureChoice(p, value)
class Word(ps:FiniteMixture[Proportions], z:MixtureChoiceVariable, value:String)
   extends CategoricalMixture[String](ps, z, value)
class Document(val file:String)
   extends ArrayBuffer[Word] { var theta:DirichletMultinomial = null }

val phis = FiniteMixture(numTopics)(new GrowableDenseDirichletMultinomial(0.01))
val documents = new ArrayBuffer[Document]
for (directory <- directories) {
   for (file <- new File(directory).listFiles; if (file.isFile)) {
      val doc = new Document(file.toString)
      doc.theta = new DenseDirichletMultinomial(numTopics, 0.01)
      for (word <- lexer.findAllIn(file.mkString).map(_ toLowerCase)) {
         val z = new Z(doc.theta, random.nextInt(numTopics))
         doc += new Word(phis, z, word)
      }
      documents += doc
   }
}
Example: Latent Dirichlet Allocation

```scala
class Z(p: Proportions, value: Int) extends MixtureChoice(p, value)
class Word(ps: FiniteMixture[Proportions], z: MixtureChoiceVariable, value: String) extends CategoricalMixture[String](ps, z, value)
class Document(val file: String) extends ArrayBuffer[Word] { var theta: DirichletMultinomial = null }

val phis = FiniteMixture(numTopics)(new GrowableDenseDirichletMultinomial(0.01))
val documents = new ArrayBuffer[Document]
for (directory <- directories) {
  for (file <- new File(directory).listFiles; if (file.isFile)) {
    val doc = new Document(file.toString)
    doc.theta = new DenseDirichletMultinomial(numTopics, 0.01)
    for (word <- lexer.findAllIn(file.mkString).map(_ toLowerCase)) {
      val z = new Z(doc.theta, random.nextInt(numTopics))
      doc += new Word(phis, z, word)
    }
    documents += doc
  }
}

val sampler = new CollapsedGibbsSampler(phis ++ documents.map(_.theta))
val zs = documents.flatMap(document => document.map(word => word.choice))
sampler.processAll(zs)
```
Experimental Comparison

• Joint Segmentation & Coreference of research paper citations (*Cora data*).
  - 1295 mentions, 134 entities, 36487 tokens

• Compare with Markov Logic Networks (*Alchemy*)
  - Same observable features

• *FACTORIE* results:
  - ~25% reduction in error (*segmentation & coref*)
  - 3-20x faster

  coref results:

<table>
<thead>
<tr>
<th></th>
<th>Prec/Recall</th>
<th>F1</th>
<th>Cluster Rec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fellegi-Sunter</td>
<td>78.0/97.7</td>
<td>86.7</td>
<td>62.7</td>
</tr>
<tr>
<td>Joint MLN</td>
<td>94.3/97.0</td>
<td>95.6</td>
<td>78.1</td>
</tr>
<tr>
<td>Isolated</td>
<td>97.09/95.42</td>
<td>96.22</td>
<td>86.01</td>
</tr>
<tr>
<td>Joint</td>
<td>95.34/98.25</td>
<td>96.71</td>
<td>94.62</td>
</tr>
</tbody>
</table>
Outline

• Motivate software engineering for statistics
• Graphical models for Extraction & Integration
  - Extraction (linear-chain CRFs)
  - Information Integration (really hairy CRFs, MCMC, SampleRank)
• Probabilistic Programming: FACTORIE
• Example
  • Relation Extraction (cross-document, w/out labeled data)
• Probabilistic Programming inside a DB
• Ongoing Work
KnowledgeBase Augmentation
[Yao, Riedel, McCallum 2010]

founded(Bill Gates, Microsoft)
nationality(Steve Jobs, USA)
...(..., ...)
founded(Paul Porter, Industry Ears)
founded(D.L. Sifry, Technorati)

488K relation instances, 54 types

334K entities, 10 types

Microsoft was founded by Bill Gates
With Microsoft chairman Bill Gates soon relinquishing ...

Paul Porter, a founder of Industry Ears

Freebase

Joint model of entities & relations

KnowledgeBase Augmentation

The New York Times

2 years
216K articles
Entities and Relations

[Yao, Riedel, McCallum 2010]

This includes features that inspect the lexical

fine the following conditional distribution:

\[
\frac{1}{\theta_1, \ldots, \theta_{n-1}} \left( \frac{1}{\theta_1, \ldots, \theta_{n-1}} \right) = \log \Psi \]

To capture the correlations between entity types

∈

In order to extract relations from text1 we need to

combination of relation and entity types

that prefers certain

To be larger than

a set

founded by

founder

When the template

a weight

Joint

Inference

Microsoft was founded by Bill Gates...

With Microsoft chairman Bill Gates soon relinquishing...

Bill Gates was born in the USA in 1955

Cross-document
Relation Extraction Experiments

Training Set

Manual Evaluation

2 years

1 year

Precision @50

<table>
<thead>
<tr>
<th></th>
<th>Isolated</th>
<th>Pipeline</th>
<th>Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.78</td>
<td>0.82</td>
<td>0.94</td>
</tr>
</tbody>
</table>
Outline

• Motivate software engineering for statistics

• Graphical models for Extraction & Integration
  - Extraction  (linear-chain CRFs)
  - Information Integration  (really hairy CRFs, MCMC, SampleRank)

• Probabilistic Programming: FACTORIE

• Example

• Relation Extraction  (cross-document, w/out labeled data)

• Probabilistic Programming inside a DB

• Ongoing Work
Information Extraction into DB

Documents

Extraction & Matching

Database
Information Extraction into DB

Documents → Extraction & Matching → DB
Information Extraction into DB

Documents

Extraction & Matching

[Diagram showing documents flowing into a database]

Red, Blue, Green dots
Information Extraction into DB

Documents

Extraction & Matching

Database
Information Extraction into DB

Documents

Extraction & Matching

query

query proc

answer
Information Extraction into Pr DB

Documents

Extraction & Matching

Database
Information Extraction into Pr DB

Documents

Extraction & Matching

Database
Information Extraction into Pr DB
Information Extraction into Pr DB
Information Extraction into Pr DB

Documents

Extraction & Matching

query

query proc

answer
Information Extraction into Pr DB
The MCMC Alternative
Information Extraction into Pr DB
The MCMC Alternative

Documents
The MCMC Alternative

DB contains only one possible world at a time.
Information Extraction into Pr DB
The MCMC Alternative

Documents

Extraction & Matching

MH inference
Information Extraction into Pr DB
The MCMC Alternative

Documents
Information Extraction into Pr DB
The MCMC Alternative

Documents

Extraction & Matching

MH inference
Information Extraction into Pr DB
The MCMC Alternative

Documents

Extraction & Matching

query

MH inference

SQL answer
Information Extraction into Pr DB
The MCMC Alternative
Information Extraction into Pr DB
The MCMC Alternative

Documents

Extraction & Matching

query

MH inference

SQL answer
Information Extraction into Pr DB
The MCMC Alternative

Documents

Extraction & Matching

query

MH inference

SQL answer
Information Extraction into Pr DB
The MCMC Alternative

Documents

Extraction & Matching

query

MH inference

SQL answer
Information Extraction into Pr DB
The MCMC Alternative

Documents

Extraction & Matching

query

MH inference

SQL answer
Information Extraction into Pr DB

The MCMC Alternative

Extraction & Matching
query

Documents

MH inference

SQL answer
Information Extraction into Pr DB
The MCMC Alternative

Documents

Extraction & Matching

MH inference

SQL answer
Information Extraction into Pr DB
The MCMC Alternative

Documents

Extraction & Matching

query

MH inference

SQL answer

[Wick, McCallum, Miklau 2010]
Particle Filtering

Documents

Extraction & query
Matching

answer
Particle Filtering with compact representation

[Schultz, McCallum, Miklau 2010]

Documents

Extraction & Matching

Query

Answer
Ongoing Work

- Factored particle filtering
- Query-specific MCMC
- Inference caching wrt to $E[\text{query}]$
- Learned proposal distributions
- Bayesian inference in distributed systems
- Probabilistic Database of all of Wikipedia, automatically growing by reading.
Thank you!

Version 0.9 at

http://code.google.com/p/factorie