Database Principles in Information Extraction

PODS 2014 Tutorial

Benny Kimelfeld

LogicBlox

* Mentioned work was done while at IBM Research - Almaden
Outline

- Information Extraction
  - Paradigms for Information Extraction
  - Formal Framework for Rule Systems
  - Concluding Remarks
What is Information Extraction (IE)?

“Information Extraction (IE) is the name given to any process which selectively structures and combines data which is found, explicitly stated or implied, in one or more texts. The final output of the extraction process varies; in every case, however, it can be transformed so as to populate some type of database.”

J. Cowie and Y. Wilks., Handbook of Natural Language Processing, 2000

“Information extraction is the identification, and consequent or concurrent classification and structuring into semantic classes, of specific information found in unstructured data sources, such as natural language text, making the information more suitable for information processing tasks.”


In short,

data-in-text → data-in-db
From September 1936 to July 1938, Turing spent most of his time studying under Church at Princeton University. In June 1938, he obtained his PhD from Princeton.
From September 1936 to July 1938, Turing spent most of his time studying under Church at Princeton University. In June 1938, he obtained his PhD from Princeton.
Popular IE Tasks

• Named Entity Recognition
• Relation Extraction
• Temporal IE

From September 1936 to July 1938, Turing spent most of his time studying under Church at Princeton University. In June 1938, he obtained his PhD from Princeton.
Popular IE Tasks

- Named Entity Recognition
- Relation Extraction
- Temporal IE
- Event Extraction

From September 1936 to July 1938, Turing spent most of his time studying under Church at Princeton University. In June 1938, he obtained his PhD from Princeton.
Popular IE Tasks

- Named Entity Recognition
- Relation Extraction
- Temporal IE
- Event Extraction
- Coreference Resolution

*From September 1936 to July 1938, Turing spent most of his time studying under Church at Princeton University. In June 1938, he obtained his PhD from Princeton.*
IE Everywhere...

Semantic Search
Semantic understanding & indexing of content to better match user's intent

Google, Bing, Yummly, PubMed, DBLife, CiteSeer

e-Commerce
Compassion Shopping extracts products and prices from multiple resources

Log Analysis
Summarize, visualize and analyze logs produced by machines

CRM / BI
Monitor customer communication and online posting for sentiment & business leads

XPLG, SolarWinds

Homeland Security
Monitor online activity to detect threats

KANA, Attensity, RelateIQ, Clarabridge, SSAS, Oracle, Social Cloud
IE in Common Text-Analytics Flow

- **Document Collection**
- **Information Extractor**
  - **Prior knowledge base**
  - **Extracted knowledge**
  - **Join / Link / ER**
    - **product—sentiment**
    - **disease—medical text**
    - **customer—social profile**
- **Global Analysis Tool**
  - (Indexing, aggregation, statistics, prediction, pattern mining, …)
  - **facts**

**UI**
Outline

• Information Extraction

• Paradigms for Information Extraction
  • Formal Framework for Rule Systems

• Concluding Remarks
Next: Four Paradigms

**Rules**
Specify a scan-annotate program (finite state transducer) by grammar rules + declarations of relational manipulation

**Classification**
Generate extraction candidates, generate features, classify candidates into answers/non-answers

**Probabilistic Graphical Models**
Specification of a probability space: random variables + dependencies

**Soft Logic**
Like rules, but effectively define a probability space over databases

Sources include [S. Sarawagi: *Information Extraction*, 2008]
IE via Rules

1. Transducer rules: Define a (non-deterministic) finite-state automaton that scans the text and produces annotations/tuples

2. Structure rules: Process facts from the rules of the 1st type, to refine the result or produce new facts

A. Meliou, D. Suciu: Tiresias: the database oracle for how-to queries, SIGMOD 2012: 337-348

- Rules learned from examples: **rule specification**: SRV [Freitag 98], WHISK [Soderland 99]; **rule generalization**: RAPIER [Califf & Mooney 99], (LP)² [Ciravegna 99]; **inductive logic**: [Aitken 02], SystemT [Nagesh et al. 12]
- Semi-automation: **rule refinement** [Liu et al. 10], **dictionary learning** [Riloff & Jones 99] [Coden et al. 12] [Roy et al. 13]
IE via Classification

IE via Classification

- Standard process:
  - e.g., [Isozaki & Kazawa 02] [Li, Bontcheva & Cunningham 04]

1. Generate candidate tokens/spans/facts
   - For example, rules with high recall, low precision

2. Generate **features** for each candidate

3. Classify each candidate: `Function(features) = true / false`

Example: find Title tokens (Variation on [Li, Bontcheva, Cunningham 04])
IE via Graphical Models

• Hidden Markov Model (HMM)
  – [Leek 97] [Miller & Bikel 97] [Brokar et al. 01] [Wang et al. 10] ...
  – Probabilistic generative process: every label emits its token and produces the next label
  – Model (emission + transition probabilities) learned from examples
  – Typical extraction: find the most likely sequence of labels, given the tokens

A → A → A → A → A → A → T → T → T → T → T

1st Cap Plural Period English
IE via Graphical Models

- Hidden Markov Model (HMM)
- Maximum Entropy Markov Model (MEMM)
  - [McCallum et al. 00] [Klein & Manning 02] ...
  - Probabilistic generative process: every label is produced by its token and the previous label

```
M. H. A. Newman, Alan M. Turing: A Formal Theorem in
```

IE via Graphical Models

- Hidden Markov Model (HMM)
- Maximum Entropy Markov Model (MEMM)
- Markov/Conditional Random Fields
  - [Lafferty et al. 01] [Wang et al. 11] [Chen et al. 12] ...
  - “Markov property:” a node is independent of the other nodes given its neighbors (other flavors exist)
  - Hammersley–Clifford theorem: under natural conditions, such a distribution is a factorization over the cliques: $\Pr[\text{graph}] \approx \prod_{\text{cliques}} F(c)$

A A A A A T T T T
M. H. A. Newman, Alan M. Turing: A Formal Theorem in

T T T T J J J P Y
IE via Soft Logic

• Deduction driven by logical assertions ...
• ... but assertions treated as *recommendations* that can be violated
• Probabilistic logic: define a probability space over possible worlds (= collections of ground facts)
  – The probability of a possible world is defined by the level to which the logical assertions are satisfied
  – Inference: most possible world, \( \text{Pr}(\text{event}) \), \( \text{Pr}(\text{event} | \text{condition}) \)
• Applied to IE
  – **MaxSat**: *SOFIE* [Suchanek et al. 09]
  – **Probabilistic Soft Logic** [Pujara et al. 13]
  – **Markov Logic**: *Alchemy* [Poon & Domingos 07], *Tuffy* [Niu et al. 11], [Satpal et al. 11]
Example: IE via Markov Logic

(Simplistic variation on [Poon & Domingos 07]; http://alchemy.cs.washington.edu)

- Tag[i-1]='A' → Tag[i]='A'
  - cost = 3
- InDict(Txt[i], 'PersonNames') → Tag[i]='A'
  - cost = 5
- HasPunc(Txt[i-1]) → Tag[i-1]≠Tag[i]
  - cost = 3
- Tag[i]='T' ∧ Tag[j]='A' → i<j
  - cost = 2

Schema: Txt[pos]=text  Tag[pos]=tag

Example:

```
```
Example: IE via Markov Logic

(Simplistic variation on [Poon & Domingos 07]; http://alchemy.cs.washington.edu)

Tag[i-1] = 'A' → Tag[i] = 'A'  
InDict(Txt[i], 'PersonNames') → Tag[i] = 'A'  
HasPunc(Txt[i-1]) → Tag[i-1] ≠ Tag[i]  
Tag[i] = 'T' ∧ Tag[j] = 'A' → i < j

Schema:  
Txt[pos] = text  
Tag[pos] = tag

Txt[10] = 'Theorem'  
Tag[10] = 'T'

Pr[world] ≈ exp(−Σ(c))

Tht[i]=‘A’  →  Tag[i]=‘A’  
InDict(Txt[i] , ‘PersonNames’) → Tag[i]=‘A’  
HasPunc(Txt[i-1]) → Tag[i-1]≠Tag[i]  
Tag[i]=‘T’ ∧ Tag[j]=‘A’ → i<j

Tht cost violations

3  i=8  3
5  i=12  5
3  i=4  3
2  (i,j)=(8,1)  (i,j)=(9,1), ...

M. H. A. Newman, Alan M. Turing: A Formal Theorem in

Automation & Conciseness

(Simplistic variation on [Poon & Domingos 07]; http://alchemy.cs.washington.edu)

- \( \text{Tag}[i-1] = +t \rightarrow \text{Tag}[i] = +t \)
- \( \text{Tag}[i-1] = \text{‘A’} \rightarrow \text{Tag}[i] = \text{‘A’} \)
- \( \text{InDict}(\text{Txt}[i], \text{‘PersonNames’}) \rightarrow \text{Tag}[i] = \text{‘A’} \)
- \( \text{HasPunc}(\text{Txt}[i-1]) \rightarrow \text{Tag}[i-1] \neq \text{Tag}[i] \)
- \( \text{Tag}[i] = \text{‘T’} \land \text{Tag}[j] = \text{‘A’} \rightarrow i < j \)

Macro: separate rule (+cost) for each value of \( +t \)
(can be used to, e.g., concisely specify a HMM)

Costs can be learned from examples!
Using Deep Parsing

- Deep parsing can provide highly valuable input for IE
- Can be incorporated in all paradigms:
  - Rules & soft logic: additional relations to consult
  - Classification: additional features (also, “graph kernels”)
  - Graphical models: additional connections/edges (CRF)
- Caveats:
  - May be computationally expensive; IE can often do w/o full parsing
  - Provides another source of imprecision
  - May mismatch dialect/slang (e.g., train on WSJ, run on FB)
    - *Text normalization* to bridge the gap [Zhang, Baldwin, Ho, K, Li, ACL’13]
Rules vs. Statistics, Industry vs. Academia

[Chiticariu, Li, Reiss, EMNLP’ 13]

- Entity extraction
- 54 industrial vendors (Who’s Who in Text Analytics, 2012)

Why so?
“[…] rules are effective, interpretable, and are easy to customize by non-experts to cope with errors.”

Gupta & Manning, CONLL 2014
Room for Both

Statistical Solution

Rule System

Feature Engineering

Cleaning + Post Proc.

Model Space, Runtime

Cleaning + Post Proc.

Building blocks
(e.g., dictionaries, NER)

“What doesn’t work: Anything requiring high precision and full automation”

Feldman & Ungar, KDD’08 tutorial on text mining
Outline

- Information Extraction
- Paradigms for Information Extraction
- Formal Framework for Rule Systems
- Concluding Remarks
Examples of IE Dev. Systems

Next, 3 IE dev. systems to motivate the formal framework: Xlog, Instaread, SystemT
Xlog: Datalog for IE

[Shen, Doan, Naughton, Ramakrishnan, VLDB 2007]

• Extension of Non-recursive (NR) Datalog
• Use case: DBLife (dblife.cs.wisc.edu)
• Data types: string, document, span
  – Focus on single-document programs
• “Procedural predicates” (p-predicates) are user-defined functions that produce relations over spans
  – Example: sentence(doc, span)
• Query-plan optimization

Kaspersky Lab CEO Eugene Kaspersky said Intel CEO Paul Otellini and the Intel board had no idea what they were in for when the company announced it was acquiring McAfee on August 19, 2010.
Xlog Example

[Shen, Doan, Naughton, Ramakrishnan, VLDB 2007]

people(d,personMention) :- docs(d), personPatterns(personPattern),
match(d, personPattern, personMention)

conferences(d,conferenceMention) :- docs(d), confPatterns(confPattern),
match(d, confPattern, conferenceMention).

chairType(d,chairType,chairPosition) :- docs(d), chairTypePatterns(chairTypePattern),
match(d, chairTypePattern, chairType),
match(d, "(?i)(vice\W+)?(co-)?chair", chairPosition),
isBefore(chairType, chairPosition),
distChar(chairType, chairPosition) < 20.

chair(d,personMention,conferenceMention,chairPosition,chairType) :-
people(d, personMention), conferences(d, conferenceMention),
chairType(d, chairType, chairPosition),
isBefore(conferenceMention, chairType),
isBefore(chairPosition, personMention),
distChar(chairPosition, personMention) < 20.

Figure 3: A sample Xlog program in our experiments.

“Declarative Information Extraction using Datalog with Embedded Extraction Predicates”
Instaread: Datalog + NLP

[Hoffmann, 2012]

• Datalog syntax
  – Types: string, span

• Built in collection of p-predicates
  – Various types of built-in regex formulas

\[
\text{killed}(a, c) \iff \text{next}(a, b) \land \text{next}(b, c) \land \text{token}(b, \text{'killed'})
\]

\[
\text{Hoffmann's dissertation} \land \text{capitalized}(a) \land \text{capitalized}(b)
\]

– Linguistic: deep parsing, coreference resolution, named-entity extractor

For an illustrative example of an extraction rule, let us assume we would like to extract instances of the killed(killer,victim) relation from text. We could create the following rule:
SystemT: SQL for IE

• Engine for AQL: SQL-like declarative language for IE
  – AQL = Annotation Query Language

• SystemT = AQL + Runtime + Dev. Tooling
  – [Chiticariu et al., ACL 2010]: establishment of SystemT as a high-quality and high-efficiency IE solution
  – System and IDE demos in ACL 2011, SIGMOD 2011

• Commercial product, high academic presence
  – Integration on public financial records [Hernández et al., EDBT’ 13, Balakrishnan et al. SIGMOD’ 10], NER [Chiticariu et al. EMNLP’ 10, ACL’ 10, Nagesh et al. EMNLP’ 12, Roy et al. SIGMOD’ 13], IR [Zhu et al. WWW’ 10, K et al. SIGIR’ 12, CIKM’ 12], sentiment analysis [Hu et al., Interact’ 13], social media [Sindhwani et al., IBM Journal 2011]
SystemT’s AQL Example

create view Caps as
extract regex /[A-Z](\w|-)+/ on D.text as name from Document D;

create view Last as
extract dictionary LastGaz on D.text as name from Document D;

create view CapsLast as
select CombineSpans(C.name, L.name) as name
from Caps C, Last L
where FollowsTok(C.name, L.name, 0, 0);

... regex + join w/ previous views

create view PersonAll as
(select R.name from FirstLast R) union all ...
projection ... union all (select R.name from CapsLast R);

create view Person as select * from PersonAll R
consolidate on R.name using 'ContainedWithin';

output view Person;

[Chiticariu, Krishnamurthy, Li, Raghavan, Reiss, Vaithyanathan, ACL 2010]
DB Research Motivated by SystemT

• How to automatically learn/improve rules?
  – Inductive logic [Nagesh et al., SIGMOD’12]
  – Provenance [Liu et al., PVLDB’10]

• How to refine the underlying dictionaries?
  – Provenance [Roy et al., SIGMOD’13]
  – Deletion propagation [K et al., PODS’11, PODS’12, TODS’12, PVLDB’13]

• How to identify recurring structures in examples?
  – Frequent graph mining [K & Kolaitis, PODS’13]

• How to formally reason about the QL?
  – Spanner theory [Fagin et al., PODS’13]
  – Cleaning & repairs [Fagin et al., PODS’14]
Theoretical Framework

• Repeated concept: Extend a relational query language with text transducers (p-predicates, usually regex formulas)
• Research challenge: theoretical underpinnings of this combined document/relational model
• Expressive power
  – Query-plan optimization: _Can we rewrite an operator via “easier” building blocks?_
  – System extensions: _Can we express a new operation using existing ones, or prove impossibility?_
Kaspersky Lab CEO Eugene Kaspersky said Intel CEO Paul Otellini and the Intel board had no idea what they were in for when the company announced it was acquiring McAfee on August 19, 2010.
**Document Spanners**

[Fagin, K, Reiss, Vansummeren, PODS 2013]

**Def:** A *document spanner* (or just *spanner* for short) is a function that maps each *document* (string) into a *relation over the spans* of that document

More formally:
- Finite alphabet $\Sigma$ of *symbols*
- A spanner maps each doc. $d \in \Sigma^*$ into a relation over the spans $[i,j)$ of $d$
- The relation has a *fixed signature* (set of attributes)
  - The attributes come from an infinite domain of *variables* $x, y, z, \ldots$

---

Kaspersky Lab CEO Eugene Kaspersky said Intel CEO Paul Otellini and the Intel board had no idea what they were in for when the company announced it was acquiring McAfee on August 19, 2010...

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,14)</td>
<td>[19,36)</td>
<td>[1,36)</td>
</tr>
<tr>
<td>[42,47)</td>
<td>[52,65)</td>
<td>[42,65)</td>
</tr>
<tr>
<td>[202,210)</td>
<td>[215,225)</td>
<td>[202,225)</td>
</tr>
</tbody>
</table>

Document $d$  

Relation over the spans of $d$
## String DB, Spanners, Interval Algebra

<table>
<thead>
<tr>
<th>String Databases</th>
<th>Spanners</th>
<th>Interval Algebra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic value: <strong>string</strong></td>
<td>Atomic value: <strong>span</strong> (pointing to doc)</td>
<td>Atomic value: <strong>interval</strong> (no text)</td>
</tr>
<tr>
<td>Join by <strong>string</strong> conditions (e.g., x is a substring of y)</td>
<td>Join by <strong>interval+string</strong> conditions (e.g., x a token in y)</td>
<td>Join by <strong>interval</strong> conditions (e.g., x is a sub-interval of y)</td>
</tr>
</tbody>
</table>

### Apps: text predicates in DBs
- [Grahne & al. 99] [Benedikt & al. 03], string manipulation
- [Bonner & Mecca 98], [Ginsburg and Wang 98]

### App: IE

<table>
<thead>
<tr>
<th>Kaspersky</th>
<th>Kaspersky</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel</td>
<td>Otellini</td>
</tr>
<tr>
<td>IBM</td>
<td>Rometty</td>
</tr>
</tbody>
</table>

### Interval Algebra

- [10, 20)
- [16, 26)
- [32, 37)
- [50, 58)
- [105, 108)
- [121, 128)

- [10, 20)
- [16, 26)
- [32, 37)
- [50, 58)
- [105, 108)
- [121, 128)
Spanners as Regex Formulas

- Regular expression with embedded variables
  \[ \gamma := \emptyset \mid \epsilon \mid \sigma \mid \gamma \lor \gamma \mid \gamma \cdot \gamma \mid \gamma^* \mid x\{\gamma\} \]

  **Ordinary regex**

  **Span variable**

- Examples:
  - .* \(x\{d\d\d\d\d\}\).*
  - .* in \(w\{\text{Alabama} \mid \text{Alaska} \mid \text{Arizona} \mid \ldots\}\).*
  - (.* \(z\{[A-Z][a-z]^{*}\}, y\{[A-Z][a-z]^{*}\}\} \).*) | …

- Restriction: each “evaluation” (parse tree) assigns one span to each variable (see [Fagin et al., PODS’13])

**Representation system for spanners**
Spanners as Datalog

- Non-recursive Datalog (NR-Datalog)
- Operate over a document (not a relational db)

\[
\begin{align*}
\text{Token}(x) &:= [(\varepsilon | .*_+) x^{[a-zA-Z]+} ( ((,V_.) | .) | \varepsilon ) ] \\
\text{State}(x) &:= \text{Token}(x), [.* x^{\{\text{Georgia | Virginia | Washington}\}.}.*] \\
\text{Cap1st}(x) &:= \text{Token}(x), [.* x^{[A-Z].}.*] \\
\text{CommaSp}(x,y,z) &:= [.* z^{x.}.} , \_ y^{.}.}.*] \\
\text{Loc}(z) &:= \text{CommaSp}(x,y,z), \text{Cap1st}(x), \text{State}(y) \\
\text{RETURN}(x,z) &:= \text{Cap1st}(x), [.*x^{.}.}^{\text{from}_y^{.}.}]*], \text{Loc}(z)
\end{align*}
\]

EDBs = Spanners!

Carter_from_Plains,_Georgia,_Washington_from_Westmoreland,_Virginia

Another representation system for spanners
Spanners as Automata

- In an accepting run, each variable opens and later closes exactly once
  ⇒ Each accepting run defines an assignment to the variables
- Nondeterministic ⇒ multiple accepting runs ⇒ multiple tuples

Another representation system for spanners
Results on Expressive Power

[Fagin, K, Reiss, Vansummeren, PODS 2013]

- In every tuple, spans overlap like balanced parentheses

- **Regular Spanners**
  - Spanners definable by var-set automata
    - Spanners definable by var-stack automata
      - Spanners definable by regex formulas
        - Hierarchical spanners
          - Spanners definable by regex formulas

Implication on various results/directions:
Imp. 1: Connection to Known Concepts

- **Connection to Recognizable Relations** [Berstel 79]
  - These are unions of cross products of regular languages
  - **THM:** *The class of regular spanners is closed under a string-selection predicate iff the predicate is a recognizable relation*

- **Connection to CRPQs** [Cruz et al. 87]
  - Conjunctive Regular Path Queries have been studied as a query language for labeled graphs
  - **THM:** *Regular spanners have the same expressive power as unions of CRPQs on paths “with marked endpoints”*
    - Up to some simple and necessary adaptation between the models

Path with marked endpoints
Imp. 2: Adding String Equality

NR Datalog w/ regex formulas

Regular Spanners

+ String-equality predicate
(+substring-of, prefix-of, ...)

Regular\textsuperscript{str=} Spanners

NameSSN(x,y) := ...
SameSSN(x1,x2) := NameSSN(x1,y1), NameSSN(x2,y2), \text{str}(y1)=\text{str}(y2)

\textit{Same string, different spans}

...application from Jane Doe, social 012-345-6789, on Mar 20th...identified as John Doe, 012-345-6789, ask us to...

\begin{tabular}{|c|c|}
\hline
x1 & x2 \\
\hline
[117,125) & [875,883) \\
(Jane Doe) & (John Doe) \\
\hline
\vdots & \vdots \\
\hline
\end{tabular}
Difference with String Equality

- *Are regular* \(^{str=}\) spanners closed under difference?
  - *Why should they? Only positive operators are used...*
  - However, regex formulas (our EDBs) can introduce “negative” operations (NFAs closed under complement)

- **Thm:** The class of regular spanners is closed under difference
- **Prop:** The class of regular \(^{str=}\) spanners is closed under string-inequality selection
- **Thm:** The class of regular \(^{str=}\) spanners is closed under string-containment selection, but then, not under non-string-containment selection!
- **Cor:** *The class of regular* \(^{str=}\) *is not closed under difference*
Imp. 3: Cleaning IE Inconsistencies

• Extractors may produce inconsistent results
  – Data artifacts
  – Developer limitations

• Rather than repairing the existing extractors, common practice is to clean (intermediate) results
  – SystemT “consolidators”
  – GATE/JAPE “controls”
  – Implicit in other rule systems, e.g., WHISK [Soderland 99]
  – Goes back to POSIX regex disambiguation
Five GATE/JAPE Controls

Sequence 12345 and sequence 12.

Document

.* x{\d\d+} .* Spanner

Context

Match

All

Screenshots from GATE UI

Context

Match

Once

Sequence 1 2 3 4 5 and sequence 1 2.

Context

Match

First

Sequence 1 2 3 4 5 and sequence 1 2.

Context

Match

Appelt
SystemT Consolidators

```
create view Caps as
extract regex /[A-Z](\w|-)/+ on D.text as name from Document D;

create view Last as
extract dictionary LastGaz on D.text as name from Document D;

create view CapsLast as
select CombineSpans(C.name, L.name) as name
from Caps C, Last L
where FollowsTok(C.name, L.name, 0, 0);
...

create view PersonAll as
  (select R.name from FirstLast R) union all ...
    ... union all (select R.name from CapsLast R);

create view Person as select * from PersonAll R
consolidate on R.name using 'ContainedWithin';

output view Person;
```

[Chiticariu, Krishnamurthy, Li, Raghavan, Reiss, Vaithyanathan, ACL 2010]

Other policies built in
Cleaning via Prioritized Repairs

• Problem: existing policies are ad-hoc; how to expose a language for user declaration?
• This PODS [Fagin, K, Reiss, Vansummeren]: spanner formalism for declarative cleaning
• Key concept: prioritized repairs
  – [Staworko, Chomicki, Marcinkowski, 2012]
• Key idea: Extend extraction programs with
  – Denial constraints: which facts are in conflict?
  – Priority declarations: preference between facts
• Captures SystemT, GATE, POSIX policies
• Next talk by Ron!
Open Problems on Spanner Theory

- Complexity
  - Evaluation, conversion between representations, ...

- Recursion (motivation: deep parsing)
  - String equality adds no expressive power!
  - Satisfiability (non-emptiness) becomes undecidable
  - Corresponding transducers?

- Theory for structured + unstructured model
  - Relations+text (Xlog), Semistructured+text (NoSQL)

- Uncertainty models
  - Rule uncertainty: elegantly extend the model and theory to notions of logic softness?
  - Text uncertainty (e.g., HMMs from OCR [K & Ré, PODS’10, JACM])
Outline

• Information Extraction
• Paradigms for Information Extraction
• Formal Framework for Rule Systems
• Concluding Remarks
Summary

• IE is everywhere, even more so nowadays
• Brief overview of IE paradigms
  – Rules, classification, probabilistic graphical models, soft logic
• Discussed a formal framework
  – Motivating systems
  – Spanner representation systems: regex formulas, Datalog, variable automata
  – Results on expressive power
  – Declarative cleaning through prioritized repairs
Last Note: Ideals to Pursue

• Intuitive & expressive query language
  ▪ E.g., Datalog + recursion + spanners

• Elegantly combine structured & NLP data

• Allow for uncertainty (e.g., soft logic)
  ▪ But as much as possible, clear and predictable

• Automation:
  ▪ Infer rules/parameters from examples
    – Preferably, intuitive rules that can actually be managed
  ▪ But realize that training data is pain point!
    – [Hoffmann et al. 11] and DeepDive [Ré et al] deploy
      “distant supervision” – don’t label text, use existing DB!