InstaRead: An IDE for IE

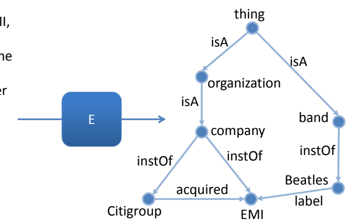
Short Presentation in CSE454

Raphael Hoffmann
January 17th, 2013

Computer Science & Engineering
UNIVERSITY of WASHINGTON

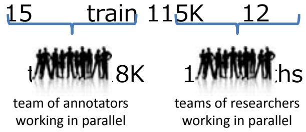
From Text To Knowledge

Citigroup has taken over EMI, the British music label of the Beatles and Radiohead, under a restructuring of its debt, EMI announced on Tuesday.



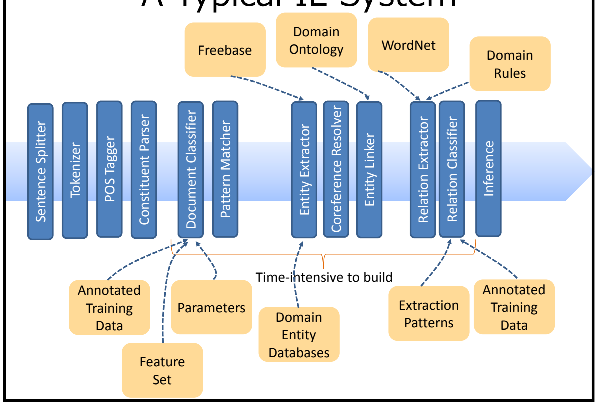
Human Effort

Evaluation	#rels	#ann words	dev time
ACE 2004	51	train 300K	12 months
MR IC 2010	15	train 115K	12 months
MR KBP 2011	16	t… 8K	1… hs

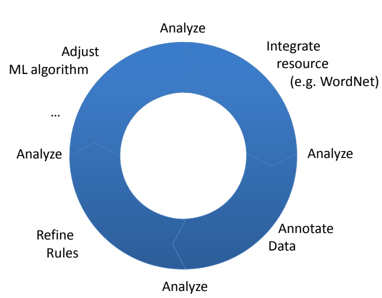


Motivates research on learning with less supervision,
but best-performing systems use more direct human input (eg. rules)

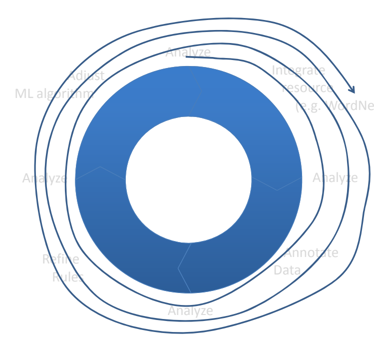
A Typical IE System



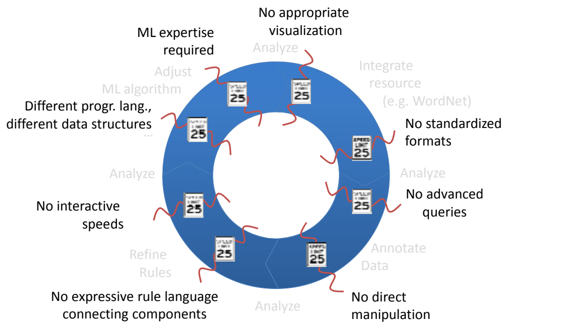
Development Cycle



Need Many Iterations

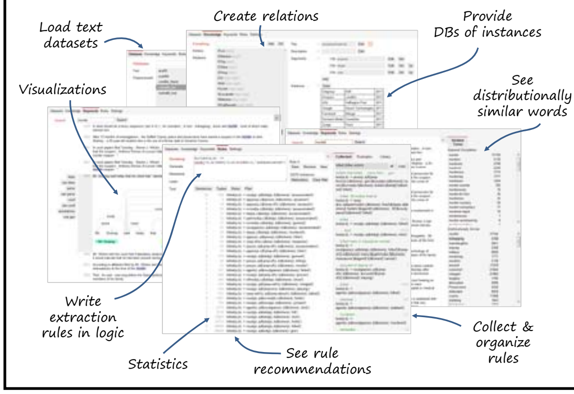


Iteration is Slow



Remove barriers!

InstaRead: An IDE for IE



Key Ideas

1. User writes rules in simple, expressive language
   Use First-Order Logic
2. User instantly sees extractions on lots of text
   Use Database Indexing
3. User gets automatic rule suggestions
   Use Bootstrapping, Learning

Demo

- Browser-based tool
- API
  - cd /projects/pardosa/s1/raphaelh/github/readr/exp
  - source ../../init.sh
  - mvn exec:java –Dexec.mainClass=newexp.Materialize

Project Ideas

In general: Create a new component & use InstaRead to explore, tie components together, and analyze

- Integrate exist. component (eg. Entity Linking, OpenIE)
- Mine rules from WordNet, FrameNet, VerbNet, Wiktionary
- Generate rule-suggestions through clustering
- Generate rule-suggestions through (better) bootstrapping
- Set rule (and thus extraction) confidence based on overlap with knowledge-base
- Develop rules/code to handle specific linguistic phenomenon (eg. time, modality, negation)
- Experiment with joint-inference of parsing + rules

More detailed slides (optional)

Rules are Crucial

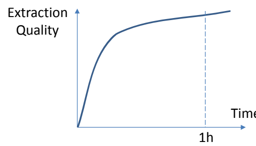
For both hand-engineered and learned systems

- Example

  r(a, b) ⇐ PER(a) ∧ nsubj(c, a) ∧ token(c, 'born') ∧ prep-in(c, b) ∧ LOC(b)

- Rules as patterns
- Rules as features
- Rules (or space of rules) typically supplied

Goal: Enable experts to write quality rules *extremely quickl*



Writing Rules in Logic

- FOL[1] is simple, expressive, extensible, widely used
- Introduce predicates for NER, dependencies, …

$$\mathrm{PER} \xleftarrow{\text{nsubj}} \text{born} \xrightarrow{\text{prep-in}} \mathrm{LOC}$$

$$\downarrow$$

r(a, b) ⇐ PER(a) ∧ nsubj(c, a) ∧ token(c, 'born') ∧ prep-in(c, b) ∧ LOC(b)

- Rules are deterministic, execute in defined order

[1] We use the subset referred to as 'safe domain-relational calculus'

Rule Composition

- Define new predicates for similar substructure

```
killingNoun('murder')
killingNoun('assassination')
killingNoun('killing')
killingNoun('slaughter')

killingOfVictim(c, b) ⇐ prep-of(c, b) ∧ token(c, d) ∧ killingNoun(d)
killingOfVictim(c, b) ⇐ nn(c, b) ∧ token(c, d) ∧ killingNoun(d)
killingOfVictim(c, b) ⇐ poss(c, b) ∧ token(c, d) ∧ killingNoun(d)

killed(a, b) ⇐ person(a) ∧ person(b) ∧ nsubjpass(c, a) ∧ token(c, 'sentenced') ∧
               prep-for(c, d) ∧ killingOfVictim(d, b)
killed(a, b) ⇐ person(a) ∧ person(b) ∧ prep-by(c, a) ∧ killingOfVictim(c, b)
```

9 instead of 24 rules!

- More compact set of rules, better generalization

Enabling Instant Execution

- Translation to SQL allows dynamic optimization

r(t) ⇐ str2syn('LeeHarveyOswald', s) ∧ syn2pos(s, p) ∧ nsubj(c, p) ∧ token(c, t)

- Each predicate maps to a fragment of SQL
- Intensional and extensional predicates
- Indices, caching, SSDs important

Experimental Setup

Datasets		Procedure
22M news sentences	½ NYTimes[1]	**Develop** • 4 relational extractors in 55min each
1M news sentences	NYTimes07[1]	**Compare** • To a weakly supervised system
22M news sentences	½ NYTimes[1]	• Bootstrap, Keyword, Morphology, and Decomposition features
5K selected sentences (some gold annotations)	CoNLL04[2]	• To gold annotations for error analysis

[1] LDC2008T19, [2] Roth & Yih, 2004

Comparison to Weakly Supervised Extraction

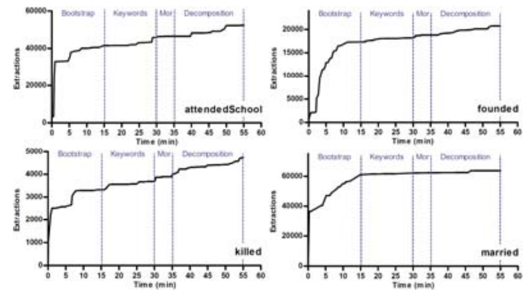
		attendedSchool	founded	killed	married
Rules	Precision	1.00	.91	.90	.90
	#extractions	1,411	997	189	4,694
Weakly supervised	Precision	0	.71	N/A	.50
	#extractions	5	14	N/A	2

- Precision consistently at least 90%
- Works well, even when weak supervision fails

Development Phases

1. Bootstrap Tool (0:15)
2. Keywords Tool (0:15)
3. Morphology Fe
   mined morphology from Wiktionary (tense etc.)
4. Decomposition Feature (0:20)
   enable chaining of rules

Comparison of Development Phases



- Bootstrap initially effective, but recall limited
- Decomposition effective for some relations

Error Analysis

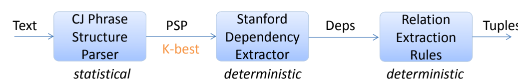
- On CoNLL04 'killed' wrt gold annotations: Re .34, Pr .98

False Negatives due to	
Preprocessing (missing predictions)	
NER	30
Dependencies	25
Co-references	7
Rules (missing predictions)	
Lexical items	89
Syntactic variation	17
Reasoning chain	10

False Positives due to	
Preprocessing (wrong predictions)	
NER	0
Dependencies	2
Co-references	0
Rules (wrong predictions)	
Lexical items	0
Syntactic variation	0
Reasoning chain	0

- 36% of all errors are due to incorrect pre-processing

Joint Parsing & Relation Extraction



- If top parse is wrong ...
  - In 35% of cases: correct at top-2
  - In 50% of cases: correct among top-5
  - In 90% of cases: correct among top-50
- Heuristic: Select first parse among top-k with most extractions

	Recall	Precision	F1
Pipeline	.342	.978	.507
Joint	.412	.973	.58

18/19 flipped to correct parse

Almost no drop in precision

Runtime Performance

	avg	median	max
#SQL queries per relation (55min)	55	54	66
#join tables per SQL query	4.3	4	11
Execution time (s) per SQL query	1.5	.074	52.5

- On 22M sentences, 3.7B rows in 75 tables, 140GB
- Most queries execute in 74ms or less
- Outliers due to Bootstrap tool
  - Aggregation over millions of rows motivates streaming or sampling

Logic

- Often Horn clauses are sufficient, but sometimes we need ¬,∀,∃
- Example 1: *founded* relation

Michael Dell <u>built</u> his first company in a dorm-room.

Mr. Harris <u>built</u> Dell <u>into</u> a formidable competitor to IBM.

$\neg(\exists d : \text{prep-into}(c, d))$

Desired conjunct

- Example 2: Integration of co-reference $\neg(\exists d : \ldots)$

"nearest noun-phrase which satisfies …"