Machine Reading & Open Information Extraction

Based on Oren Etzioni, Turing Center, University of Washington

http://turing.cs.washington.edu

Outline

1. What is Machine Reading?
2. Open Information Extraction
   a) Task definition
   b) KnowItAll project → TextRunner system
3. Ideas from KnowItAll project
4. Speculation about Machine Reading

What is Machine Reading?

Automatic, **unsupervised** understanding of text

corpus C

background beliefs B

yield a set of beliefs B'

$$R(C) + B \rightarrow B'$$

Machine Reading Builds on Past Work

- Information extraction
- Text mining
- Textual entailment

all are components of Machine Reading!

Often supervised, or semi-supervised

But “Reading” the Web is Tough

- Traditional IE is narrow
- IE has been applied to small, homogenous corpora
- No parser achieves high accuracy
- No named-entity taggers
- No supervised learning

How about semi-supervised learning?

Semi-Supervised Learning

- Few hand-labeled examples **per concept!**
- → Limit on the number of concepts
- → Concepts are pre-specified
- → Problematic for Machine Reading

- **Alternative:** Bootstrapping, self supervised methods
  - Learner discovers concepts on the fly
  - Learner automatically labels examples
  - Very different from clustering

2. Open IE Paradigm
(Banko, Cafarella, Soderland, et. al, IJCAI '07)

	Traditional IE	Open IE
Input:	Corpus + Hand-labeled Data	Corpus
Relations:	Specified in Advance	Discovered Automatically
Complexity:	O(D * **R**)	O(D)
	R relations	D documents
Text analysis:	Parser + Named-entity tagger	NP Chunker

Focus: Open IE on Web Text

Challenges

"Semantically tractable" sentences	Difficult, ungrammatical sentences
Redundancy	No labeled data
Search engines	Unreliable information
Broad scope	Diverse topics

Focus: Open IE on Web Text

Advantages	**Challenges**
"Semantically tractable" sentences	Difficult, ungrammatical sentences
Redundancy	No labeled data
Search engines	Unreliable information
Broad scope	Diverse topics

TextRunner (First Open IE System)

1. **Self-Supervised Learner**: automatically labels +/- examples & learns an extractor on basis of a small corpus sample as input (5000 sentences); The learner requires no hand-tagged data.

2. **Single-Pass Extractor:** single pass over corpus, identifying extractions in each sentence; utilizes no parser;

3. **Redundancy based assessor**: assign a prob. to each retained tuple by exploiting redundancy in text;

4. **Query Processor:** indexes extractions → enables queries at interactive speeds

Self-supervised Learning

1. Automatically labels its own training data as positive or negative
    1. Dependency analysis of each sentence
    2. Extraction and normalization of binary relations

2. Uses a Naïve Bayes classifier, which is then used by the extraction module
    1. Simple feature extraction as basis for classifier

TextRunner Extraction

- Extract Triple representing binary relation (Arg1, Relation, Arg2) from sentence.

EBay was originally founded by Pierre Omidyar.

EBay **was originally** founded by Pierre Omidyar.

(Ebay, Founded by, Pierre Omidyar)

Numerous Extraction Challenges

- **Drop non-essential info:**

  "was originally founded by" → **founded by**

- **Retain key distinctions**

X founded **by** Y **≠** X founded Y

- **Non-verb relationships**

"George Bush, president of the U.S…"

- **Synonymy & aliasing**

Albert Einstein **=** Einstein **≠** Einstein Bros.

Computation of triples

- For each parsed sentence:
- Determine base NPs, which serve as argument candidates
- For each pair of NPs, determine its connecting node in dependency tree (skeleton)
- Connecting words defines candidate relation
  - Not only verbs, but arbitrary elements

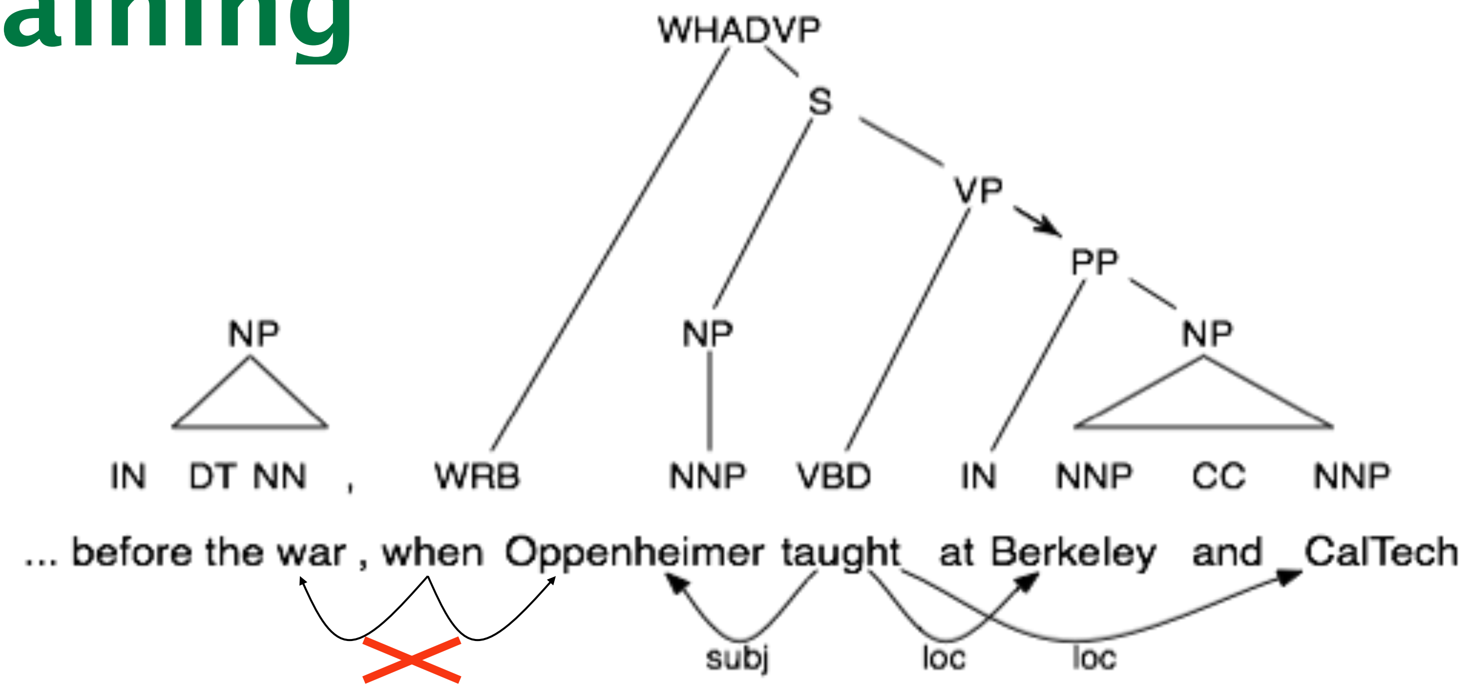
Labeling as POS or NEG

- Apply syntactic constraints, and iff all succeed, label triple as POS, else as NEG
- Some constraints:
  - Dependency chain between e_i and $e_j \leq N$
  - Path(e_i , e_j) does not cross sentence-like boundary (ako upper boundary)
  - Neither e_i nor e_j is a pronoun

Naïve Bayes Classifier

- Feature vector representation of each triple
- All feature functions are defined such that no parsing is required later, e.g.
  - POS sequence on relation $r_{i,j}$
  - # of tokens $r_{i,j}$
  - # of stop words $r_{i,j}$
  - POS tag of e = PN
  - Left/right POS of e
- NOTE: output of classifier is language specific, but does not contain relation-specific or lexical features → domain-independent

Example of Extractor Training



Instances

t1 (+) (Oppenheimer, taught at, Berkeley)
t2 (+) (Oppenheimber, taught at, CalTech)
t3 (-) (war, , when, Oppenheimer)

Features(t1)

IsProperNoun(NP1) = true
LeftContext(NP1) = WRB
Rel-Length = 2
Rel-NumStopWords = 1
Rel-StartsWith = VB
Rel-Contains(VB) = true
Rel-Contains(VB IN) = true
...

Analysis of Extraction Process

- Language-specific, but relation general!
- Parser used in training, not in extraction.
- Fast, but noisy process
  - Precision/recall can be tuned

Single-Pass Extractor

1. Triple extraction
   1. POS tagging
   2. NP chunking as basis for arguments
   3. Identify relations as text between found NPs
      1. Heuristically delete non-relevant text fragments, e.g., PPs, adverbs
2. Classification
   1. Pass each candidate triple to the classifier
   2. Labels triple as POS/NEG

Normalization of Triples

- Omit non-essential modifiers in N and V
  - "was originally developed by" → "was developed by"
- Merge and count identical normalized triples
- Assign probability value to each triple
  - Estimate probability that triple is correct instance given that it was extracted from K different sentences

Probability	Count	Arg1	Predicate	Arg2
0.98	59	Smith	invented	the margherita
0.97	49	Al Gore	invented	the Internet
0.97	44	manufacturing plant	first invented	the automatic revolver
0.97	41	Alexander Graham Bell	invented	the telephone
0.97	36	Thomas Edison	invented	light bulbs
0.97	29	Eli Whitney	invented	the cotton gin
0.96	23	C. Smith	invented	the margherita
0.96	19	the Digital Equipment Corporation manufacturing plant	first invented	the automatic revolver
0.96	18	Edison	invented	the phonograph

http://www.cs.washington.edu/research/textrunner/

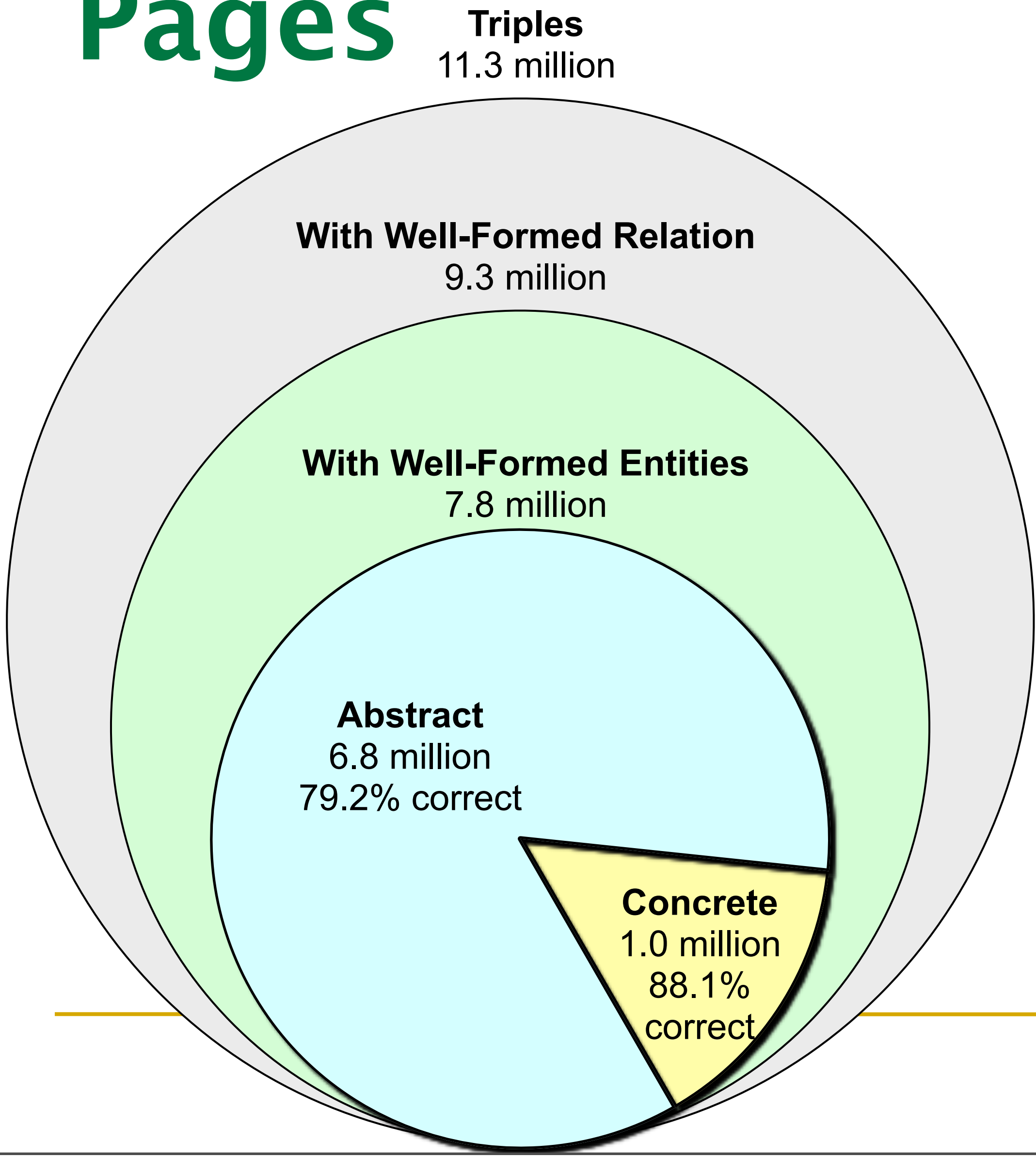
Probability	Count	Arg1	Predicate	Arg2
0.98	59	Smith	invented	the margherita
0.97	49	Al Gore	invented	the Internet
0.97	44	manufacturing plant	first invented	the automatic revolver
0.97	41	Alexander Graham Bell	invented	the telephone
0.97	36	Thomas Edison	invented	light bulbs
0.97	29	Eli Whitney	invented	the cotton gin
0.96	23	C. Smith	invented	the margherita
0.96	19	the Digital Equipment Corporation manufacturing plant	first invented	the automatic revolver
0.96	18	Edison	invented	the phonograph

http://www.cs.washington.edu/research/textrunner/

Estimating the Correctness of Facts

- Random selection of N (=400) triples
- Manual evaluation (by 3 authors)
- First, check well-formdness of relation r:
  - ∃ pair X, Y s.t. (X, r, Y) is valid
  - (FCI, specializes in, software development) but not (demands, of securing, border)
- Second, check whether X & Y are reasonable
- Subdivide relations into concrete and abstract facts
  - Basically if arguments are NEs
  - (Tesla, invented, coil transformer) but not (Einstein, developed, theory) or (executive, hired by, company)

Triples from 9 million Web Pages



- **Concrete facts:** (Oppenheimer, taught at, Berkeley)
- **Abstract facts:** (fruit, contain, vitamins)

TextRunner's Run time

- KnowItAll—runtime is linear in R
  - 10 relations ➔ ~ 3 days (types not instances !)

- TextRunner—runtime is constant in R
  - 10^3—10^5 relations ➔ ~ 3 days
  - **two to four orders of magnitude boost!**

- On the 10 relations, comparable recall but 33% fewer errors for TextRunner

“Modular” Open IE Ideas

Exploit massive and redundant corpus to

1. Keep it simple
2. The world may be flat, but text ain't
3. One thousand points of light
4. Where there's smoke, there's fire

Compensate for weaker inputs!

1. Keep it Simple

- Paris, which has been proclaimed by many literary figures to be a great source of inspiration, is also a capital city, but not the capital city of an ordinary country, but rather the capital of the great republic that we love---the republic of France!

Paris is the Capital of France.

2. World May be Flat but Text Ain't

Recover relations from text (cf. Pantel & Lin '01)

Resolver (Yates & Etzioni, HLT '07): determines synonymy based on relations found by TextRunner;
introduces a probabilistic relational model for predicting whether two strings are co-referential based on the similarity of the assertions containing them.

- (X, born in, 1941) (M, born in, 1941)
- (X, citizen of, US) (M, citizen of, US)
- (X, friend of, Joe) (M, friend of, Joe)

P(X = M) ~ shared relations

Relation Synonymy

- (1, R, 2)
- (2, R 4)
- (4, R, 8)
- Etc.

- (1, R' 2)
- (2, R', 4)
- (4, R' 8)
- Etc.

P(R = R') ~ shared **argument** pairs

•Unsupervised probabilistic model (similarity of relation strings & similarity of the assertions they appear in)
•Mutual recursion (merging classes)

Relation Synonymy in TextRunner

- How many triples are reformulations of others?
  - Which relations are synonymous?
  - Which entities are referred to mutual names?
  - Truly synonymous relations are rare to find and mostly needs domain-specific type checking, e.g., “developed” could mean “invented” or “created” depending on type of arguments

Relation Synonymy in TextRunner

- Approximate heuristics used in TextRunner
  - Merge triples on basis of leading stop words, e.g., "that are consistent with" = "which is consistent with"
  - Merge triples on basis of active/passive voice, e.g., "invented" = "is invented"

Relation Synonymy in TextRunner

- Simple experiment:
  - Cluster facts on basis of same arguments -> manual checking reveals that only 1/3 of the tuples belong to synonymy clusters
  - Computation of synonymy clusters (using heuristics above), manually analysis of 100 randomly selected clusters seem to indicate that 92% of the 11 M tuples describe distinct facts/assertions

3. One Thousand Points of Light

Illuminating Phrases reveal semantics

- Class Membership (**X** and other ***C***)

(**Hearst '92**)

Which adjective is stronger ***A*** or **B**?

- **Opine** (**Popescu & Etzioni**, **EMNLP '05**)):
  - "...clean but not spotless"
  - "very clean, almost spotless"

Resolver: is **X** = **Y**?

- "**X** and **Y**" → **X** ≠ **Y**
  - "...Hillary Clinton and Bill Clinton..."

4. Where There's Smoke There's..

Distributional Hypothesis: "words that occur in the same contexts tend to have similar meanings " (Harris, 1954)

[Brin, 1998; Riloff & Jones 1999; Agichtein & Gravano, 2000; Pasca et al. 2006; Pantel et al. 2006]

KnowItAll Hypothesis: extractions that occur in the same informative contexts **more frequently** are **more likely** to be correct.

Count Phrase Co-occurrence (Turney '02)

…<X> and other cities…

- PMI as measure of co-occurrence
- PMI via search engine hit counts
  - Query Google with phrase
    - "Atlanta and other cities"
- Yields useful semantic information

$$\text{coordination score}(s_1, s_2) = \frac{\text{hits}(s_1 \text{ and } s_2)^2}{\text{hits}(s_1) \times \text{hits}(s_2)}$$

Probabilistic Model of Redundancy
(Downey, Soderland, Etzioni, IJCAI '05)

1) Repetition (KnowItAll Hypothesis)
2) Distinct illuminating phrases

Phrase	Hits

Goal: a formal model of these intuitions

Probabilistic Model of Redundancy

(Downey, Soderland, Etzioni, IJCAI '05)

1) Repetition (KnowItAll Hypothesis)

2) Distinct illuminating phrases

Phrase	Hits
"**Atlanta** and other cities"	980

Goal: a formal model of these intuitions

Probabilistic Model of Redundancy

(Downey, Soderland, Etzioni, IJCAI '05)

1) Repetition (KnowItAll Hypothesis)

2) Distinct illuminating phrases

Phrase	Hits
"Atlanta and other cities"	980
"Canada and other cities"	286

Goal: a formal model of these intuitions

Probabilistic Model of Redundancy

(Downey, Soderland, Etzioni, IJCAI '05)

1) Repetition (KnowItAll Hypothesis)

2) Distinct illuminating phrases

Phrase	Hits
"**Atlanta** and other cities"	980
"**Canada** and other cities"	286
"cities such as **Atlanta**"	5860

Goal: a formal model of these intuitions

Probabilistic Model of Redundancy

(Downey, Soderland, Etzioni, IJCAI '05)

1) Repetition (KnowItAll Hypothesis)

2) Distinct illuminating phrases

Phrase	Hits
"**Atlanta** and other cities"	980
"**Canada** and other cities"	286
"cities such as **Atlanta**"	5860
"cities such as **Canada**"	7

Goal: a formal model of these intuitions

Problem Statement

If an extraction x appears k times in a set of n distinct sentences that match phrases suggestive of membership in C, what is the probability that $x \in C$?

C is a class ("cities") or a relation ("mayor of")

Note: we only count distinct sentences!

Noisy-Or Model (Single Phrase)

Phrase: “Cities such as”

Cities, n = 10	k
New York	2
Tokyo	2
Seattle	1
Africa	1
Paris	1
Tel Aviv	1
Kabul	1
Sydney	1

Noisy-Or Model (Single Phrase)

Phrase: "Cities such as"

Cities, n = 10	k
New York	2
Tokyo	2
Seattle	1
Africa	1
Paris	1
Tel Aviv	1
Kabul	1
Sydney	1

Noisy-Or Model:

p is the phrase's accuracy.

$p = 0.9$

Noisy-Or Model (Single Phrase)

Phrase: "Cities such as"

Cities, n = 10

	k
New York	2
Tokyo	2
Seattle	1
Africa	1
Paris	1
Tel Aviv	1
Kabul	1
Sydney	1

Noisy-Or Model:

p is the phrase's accuracy.
$p = 0.9$

$$P_{noisy-or}\left(x \in C \mid x \text{ appears } k \text{ times}\right)$$
$$= 1 - (1 - p)^k$$

Noisy-Or Model (Single Phrase)

Phrase: "Cities such as"

Cities, n = 10

	k	$P_{noisy-or}$
New York	2	0.99
Tokyo	2	0.99
Seattle	1	0.9
Africa	1	0.9
Paris	1	0.9
Tel Aviv	1	0.9
Kabul	1	0.9
Sydney	1	0.9

Noisy-Or Model:

p is the phrase's accuracy.

$p = 0.9$

$$P_{noisy-or}\left(x \in C \,\middle|\, x \text{ appears } k \text{ times}\right) = 1 - (1 - p)^k$$

Noisy-Or Model (Single Phrase)

Phrase: "Cities such as"

n = 50,000	k	$P_{noisy-or}$
New York	2	0.99
Tokyo	2	0.99
Seattle	1	0.9
Africa	1	0.9
Paris	1	0.9
Tel Aviv	1	0.9
Kabul	1	0.9
Sydney	1	0.9

Noisy-Or Model:

p is the phrase's accuracy.

$p = 0.9$

$$P_{noisy-or}\left(x \in C \,\middle|\, x \text{ appears } k \text{ times}\right) = 1 - (1 - p)^k$$

Noisy-Or Model (Single Phrase)

Phrase: "Cities such as"

n = **50,000**

	k	$P_{noisy-or}$
New York	**2**	**0.99**
Tokyo	**2**	**0.99**
Seattle	**1**	**0.9**
Africa	**1**	**0.9**
Paris	**1**	**0.9**
Tel Aviv	**1**	**0.9**
Kabul	**1**	**0.9**
Sydney	**1**	**0.9**

Noisy-Or Model:

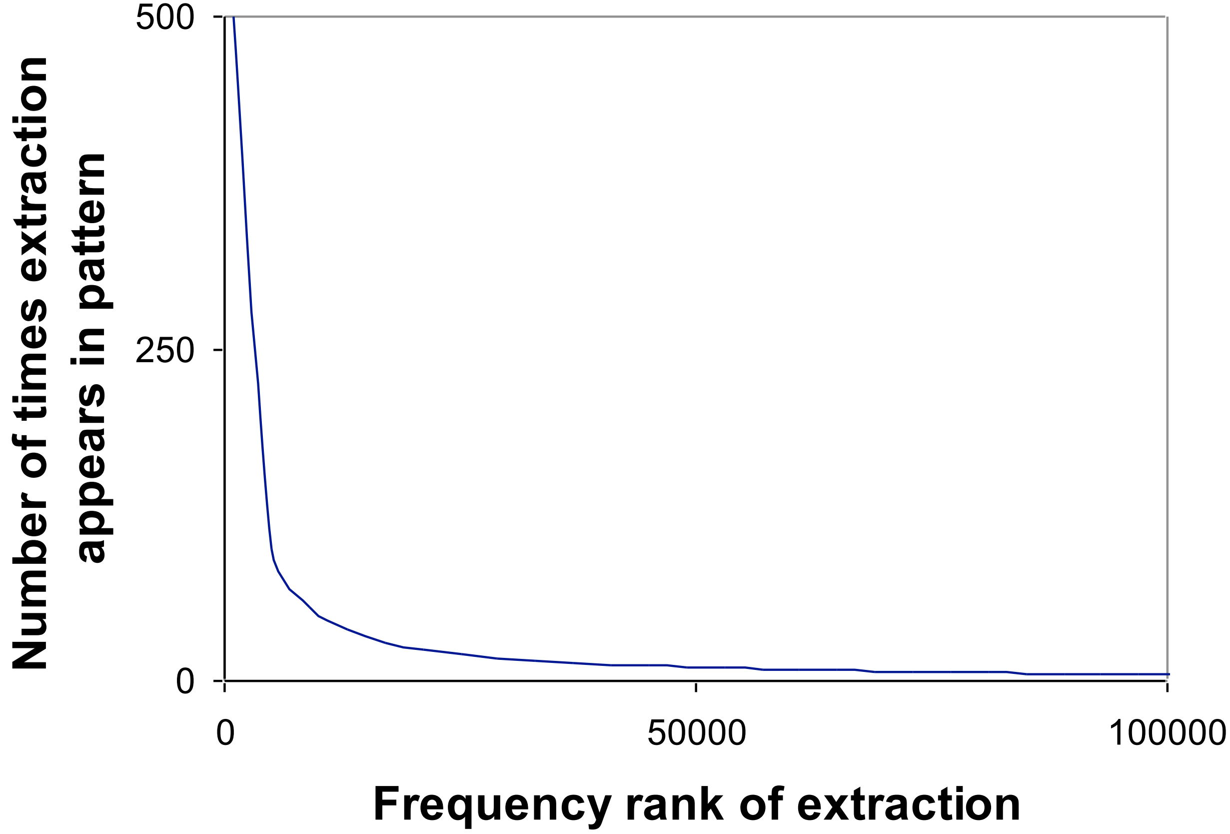
p is the phrase's accuracy.

$p = 0.9$

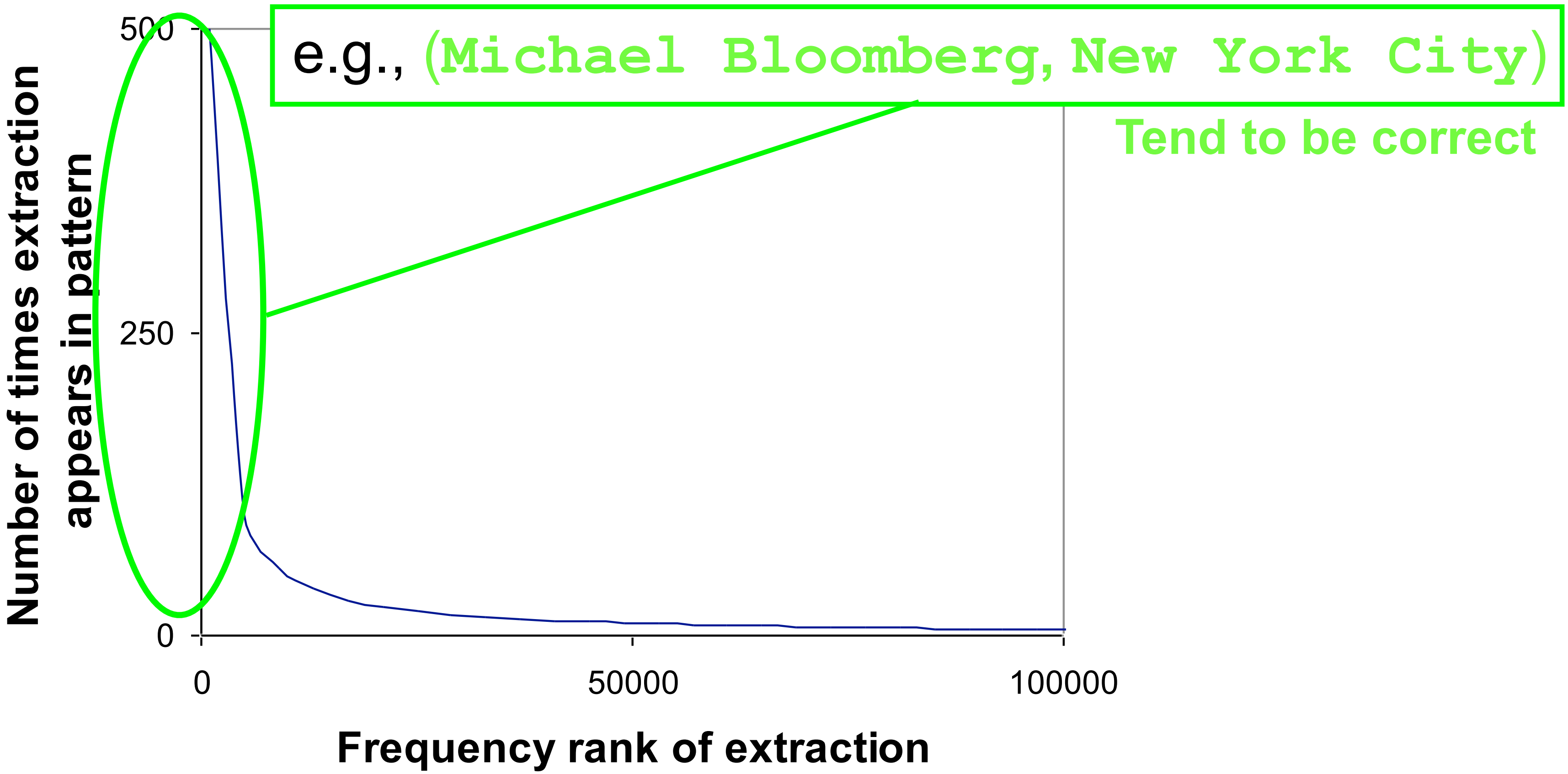
$$P_{noisy-or}\left(x \in C \mid x \text{ appears } k \text{ times}\right) = 1 - (1-p)^k$$

Noisy-or model is linear in # features

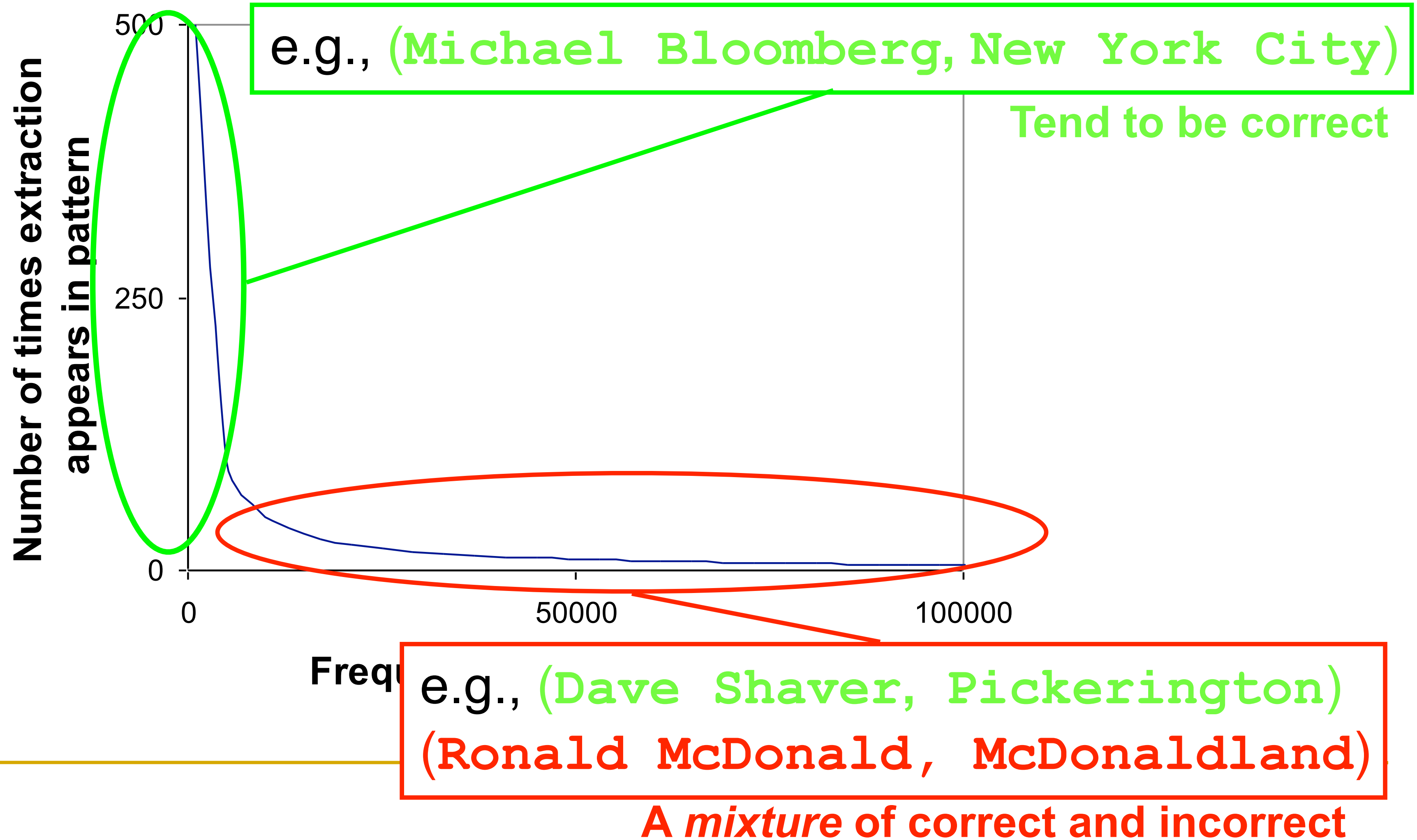
The Problem of Sparse Extractions



The Problem of Sparse Extractions



The Problem of Sparse Extractions



5. Language Models to the Rescue

Instead of illuminating phrases, leverage **all** contexts of a particular word

REALM (Downey, Schoenmackers, Etzioni, ACL '07)

Contexts captured via HMM + n-grams models

- Self supervised
- Scalable (built once per corpus)
- Boosts TextRunner's precision

Argument "Type checking" via HMM

Relation's arguments are "typed":

(Person, Mayor Of, City)

Training: Model distribution of Person & City contexts in corpus (Distributional Hypothesis)

Query time: Rank sparse triples by how well each argument's context distribution matches that of its type

Simple Example

- (Shaver, Mayor of, Pickerington) over (Spice Girls, Mayor of, Microsoft)

Because:

- Shaver's contexts are more like Giuliani's than Spice Girls', and
- Pickerington's contexts are more like Miami's than Microsoft's

Utilizing HMMs to Check Types

Challenges:

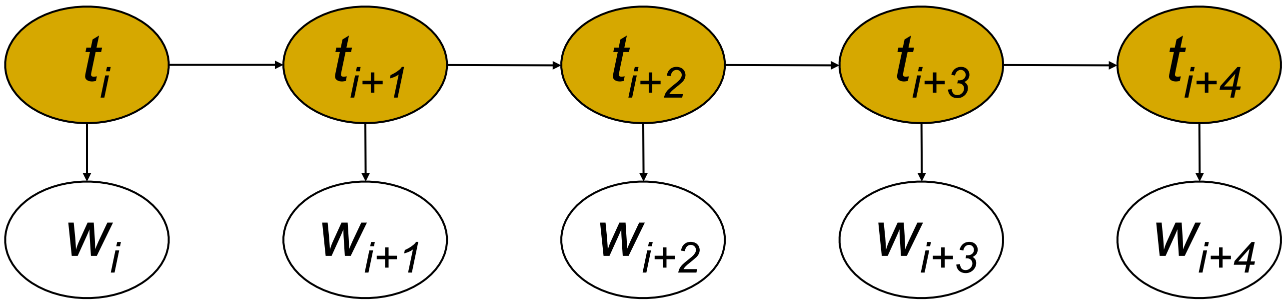
- Argument types are not known
- Can't build model for each argument type
- Textual types are fuzzy

Solution: Train an HMM for the corpus using EM & bootstrap

HMM in more detail

Training: seek to maximize probability of corpus **w** given latent states **t** using EM:

$$P(\mathbf{w}|\mathbf{t}) = \prod_i P(w_i|t_i)P(t_i|t_{i-1},\ldots,t_{i-k})$$



$$t_i \in \{1,\ldots,N\}, k = 1$$
$$w_i \in words$$

Using the HMM at Query Time

- Given a set of extractions (**Arg1**, **Rln**, **Arg2**)
- Seeds = most frequent Args for **Rln**

Using the HMM at Query Time

- Given a set of extractions (Arg1, Rln, Arg2)
- Seeds = most frequent Args for Rln

$$f(\arg, seeds) = KL\left(\frac{1}{|seeds|}\sum_i P(t \mid seed_i), P(t \mid \arg)\right)$$

Using the HMM at Query Time

- Given a set of extractions (Arg1, Rln, Arg2)
- Seeds = most frequent Args for Rln

$$f(\arg, seeds) = KL\left(\frac{1}{|seeds|}\sum_i P(t \mid seed_i), P(t \mid \arg)\right)$$

1. Distribution over t is read from the HMM
2. Compute KL divergence via f(arg, seeds)

Using the HMM at Query Time

- Given a set of extractions (Arg1, Rln, Arg2)
- Seeds = most frequent Args for Rln

$$f(\arg, seeds) = KL\left(\frac{1}{|seeds|}\sum_i P(t \mid seed_i), P(t \mid \arg)\right)$$

1. Distribution over t is read from the HMM
2. Compute KL divergence via f(arg, seeds)
3. For each extraction, average over Arg1 & Arg2
4. Sort "sparse" extractions in ascending order

HMM Analysis

Learning time is proportional to (corpus size $*N^{k+1}$)

N = number of latent states (N = 20)

k = HMM order (k = 3)

Too coarse for relation assessment

Headquartered(`Santa Clara, Microsoft)`

Relation assessment done via N-gram model

Improving TextRunner's Precision

"invented" ==>

Probability	Count	Argument 1	Predicate	Argument 2	
0.98	59	Smith	invented	the margherita	Smith invented the margherita in 1889), fo United managed to avoid antagonizing the
0.97	44	manufacturing plant	first invented	the automatic revolver	This competition takes place over a mile t Colt first invented the automatic revolver
0.97	32	Al Gore	invented	the Internet	But if he ever tells you the name was his i
0.96	27	Bell	invented	the telephone	Alexander Graham Bell : Bell invented the teaching the deaf to speak.
0.96	27	Thomas Edison	invented	light bulbs	Thomas Edison invented light bulbs.
0.96	23	C. Smith	invented	the margherita	C. Smith invented the margherita in 1889)

REALM improves precision by re-ranking

Improving TextRunner's Precision

"invented" ==>

Ranked by frequency

Probability	Count	Argument 1			
0.98	59	Smith	invented	the margherita	Smith invented the margherita in 1889), fo United managed to avoid antagonizing the
0.97	44	manufacturing plant	first invented	the automatic revolver	This competition takes place over a mile t Colt first invented the automatic revolver
0.97	32	Al Gore	invented	the Internet	But if he ever tells you the name was his i
0.96	27	Bell	invented	the telephone	Alexander Graham Bell : Bell invented the teaching the deaf to speak.
0.96	27	Thomas Edison	invented	light bulbs	Thomas Edison invented light bulbs.
0.96	23	C. Smith	invented	the margherita	C. Smith invented the margherita in 1889)

REALM improves precision by re-ranking

Improving TextRunner: Example (1)

"conquered" Top 10:

Great, Egypt
conquistador, Mexico
Normans, England
Arabs, North Africa
Great, Persia
Romans, part
Romans, Greeks
Rome, Greece
Napoleon, Egypt
Visigoths, Suevi Kingdom

TR Precision: 60%

Improving TextRunner: Example (1)

"conquered" Top 10:

Arabs, Rhodes
Arabs, Istanbul
Assyrians, Mesopotamia
Great, Egypt
Assyrians, Kassites
Arabs, Samarkand
Manchus, Outer Mongolia
Vandals, North Africa
Arabs, Persia
Moors, Lagos

TR Precision: 60% **REALM Precision: 90%**

Improving TextRunner: Example (2)

"headquartered" Top 10:

company, Palo Alto
held company, Santa Cruz
storage hardware and software, Hopkinton
Northwestern Mutual, Tacoma
1997, New York City
Google, Mountain View
PBS, Alexandria
Linux provider, Raleigh
Red Hat, Raleigh
TI, Dallas

TR Precision: 40%

Improving TextRunner: Example (2)

"headquartered" Top 10:

Tarantella, Santa Cruz
International Business Machines Corporation, Armonk
Mirapoint, Sunnyvale
ALD, Sunnyvale
PBS, Alexandria
General Dynamics, Falls Church
Jupitermedia Corporation, Darien
Allegro, Worcester
Trolltech, Oslo
Corbis, Seattle

TR Precision: 40% **REALM Precision: 100%**

Language Modeling & Open IE

- REALM improves precision@10 by 90%
- Self supervised
- Illuminating phrases → full context
  - More "efficient" than Urns!
  - Handles sparse extractions

- N-dimensional projection generalizes
  - Chicago, Illinois → Pickerington, Ohio

Conclusions

- Machine Reading is self supervised
- Open IE scales IE towards Machine Reading
- Machine Reading ≠ Human reading