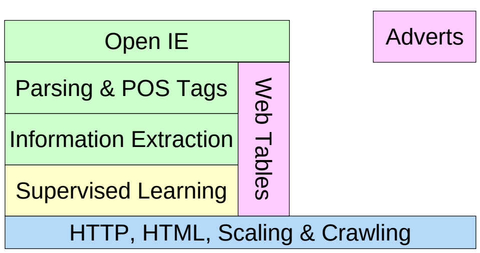
Open Information Extraction

CSE 454
Daniel Weld

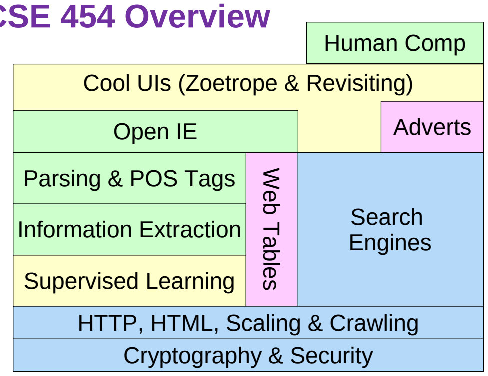
To change

- More textrunner,
- more pattern learning
- Reorder:
  - Kia start

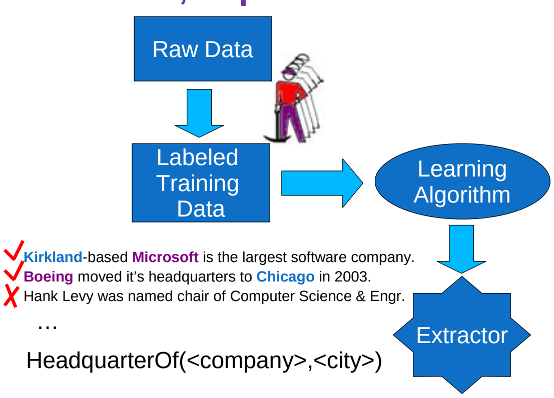
CSE 454 Overview



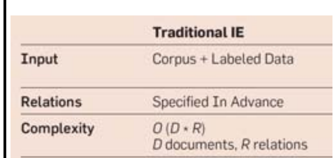
CSE 454 Overview



Traditional, Supervised I.E.



What is Open Information Extraction?



	Traditional IE
Input	Corpus + Labeled Data
Relations	Specified In Advance
Complexity	$O(D * R)$ D documents, R relations

What is Open Information Extraction?

	Traditional IE	Open IE
Input	Corpus + Labeled Data	Corpus + Domain-Independent Methods
Relations	Specified In Advance	Discovered Automatically
Complexity	$O(D * R)$ D documents, R relations	$O(D)$ D documents

Methods for Open IE

- **Self Supervision**
  - Kylin (Wikipedia)
  - Shrinkage & Retraining
  - Temporal Extraction
- **Hearst Patterns**
  - PMI Validation
  - Subclass Extraction
- **Pattern Learning**
- **Structural Extraction**
  - List Extraction & WebTables
  - TextRunner

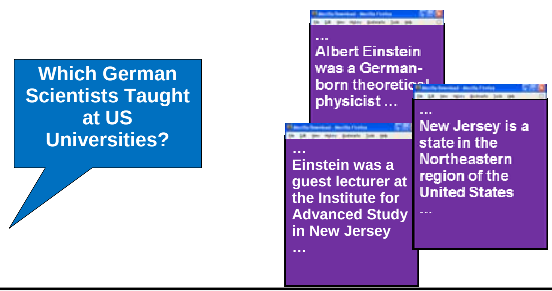
The Intelligence in Wikipedia Project

Daniel S. Weld
Department of Computer Science & Engineering
University of Washington
Seattle, WA, USA

Joint Work with
Fei Wu, Raphael Hoffmann, Stef Schoenmackers,
Eytan Adar, Saleema Amershi, Oren Etzioni,
James Fogarty, Chloe Kiddon,
Shawn Ling & Kayur Patel

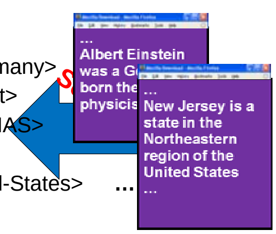
Motivating Vision

Next-Generation Search = Information Extraction
+ Ontology
+ Inference



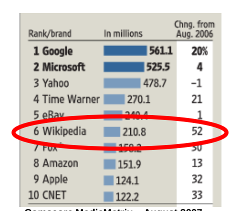
Next-Generation Search

- Information Extraction
  - <Einstein, Born-In, Germany>
  - <Einstein, ISA, Physicist>
  - <Einstein, Lectured-At, IAS>
  - <IAS, In, New-Jersey>
  - <New-Jersey, In, United-States> …
- Ontology
  - Physicist (x) ➔ Scientist(x) …
- Inference
  - Einstein = Einstein …



Why Wikipedia?

- **Comprehensive**
- **High Quality**
  [Giles Nature 05]
- **Useful Structure**
  Unique IDs & Links
  Infoboxes
  Categories & Lists
  First Sentence
  Redirection pages
  Disambiguation pages
  Revision History
  Multilingual



Comscore MediaMetrix – August 2007

Cons

Natural-Language
Missing Data
Inconsistent
Low Redundancy

Kylin: Self-Supervised Information Extraction from Wikipedia

[Wu & Weld CIKM 2007]

From infoboxes to a training set

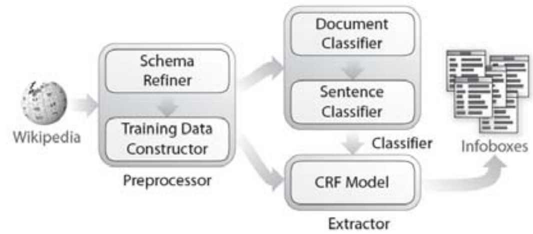
Clearfield County was created in 1804 from parts of Huntingdon and Lycoming Counties but was administered as part of Centre County until 1812.

Its county seat is Clearfield.

2,972 km² (1,147 mi²) of it is land and 17 km² (7 mi²) of it (0.56%) is water.

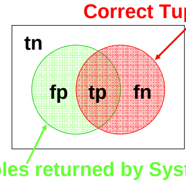
As of 2005, the population density was 28.2/km².

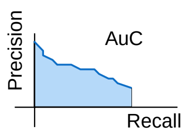
Kylin Architecture



The Precision / Recall Tradeoff

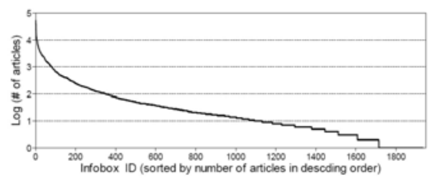
- **Precision** $\frac{tp}{tp+fp}$
  - Proportion of selected items that are correct
- **Recall** $\frac{tp}{tp+fn}$
  - Proportion of target items that were selected
- **Precision-Recall curve**
  - Shows tradeoff





Preliminary Evaluation

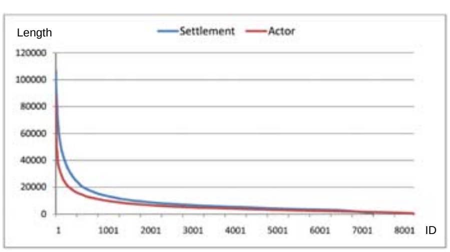
- Kylin Performed Well on Popular Classes:
  Precision: mid 70% ~ high 90%
  Recall: low 50% ~ mid 90%
- ... Floundered on Sparse Classes – Little Training Data



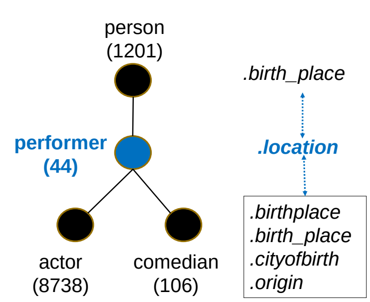
82% < 100 instances; 40% <10 instances

Long-Tail 2: Incomplete Articles

- Desired Information Missing from Wikipedia
  800,000/1,800,000(44.2%) *stub* pages *[July 2007 of Wikipedia]*

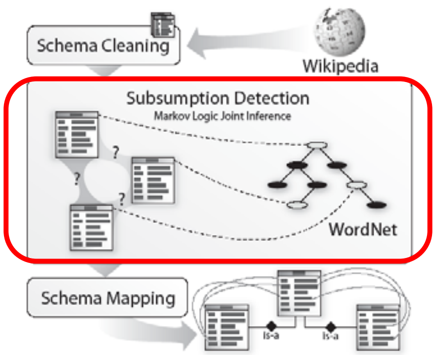


Shrinkage?



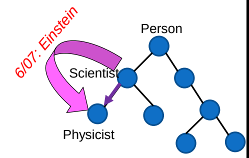
KOG: Kylin Ontology Generator

[Wu & Weld, WWW08]

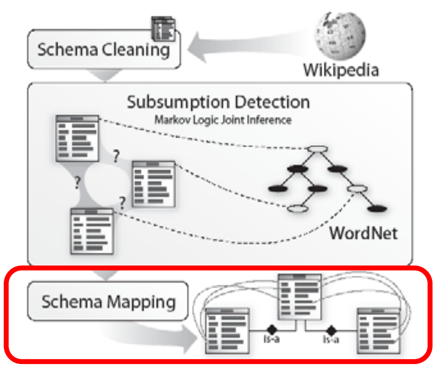


Subsumption Detection

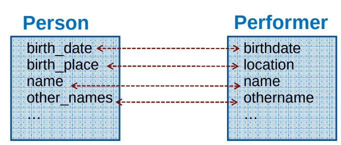
- **Binary Classification Problem**
- **Nine Complex Features**
  E.g., String Features
  … IR Measures
  … Mapping to Wordnet
  … Hearst Pattern Matches
  … Class Transitions in Revision History
- **Learning Algorithm**
  SVM & MLN Joint Inference



KOG Architecture



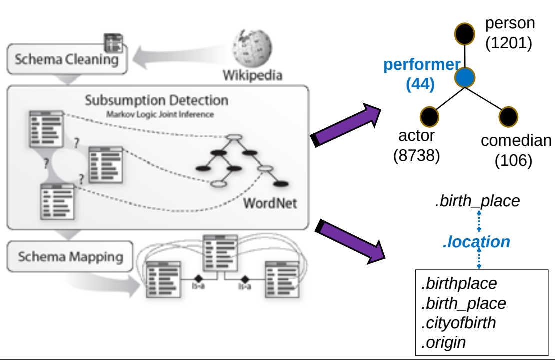
Schema Mapping



- **Heuristics**
  - Edit History
  - String Similarity
- **Experiments**
  - Precision: 94%   Recall: 87%
- **Future**
  - Integrated Joint Inference

KOG: Kylin Ontology Generator

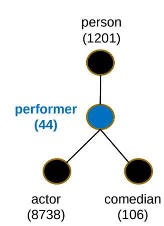
[Wu & Weld, WWW08]



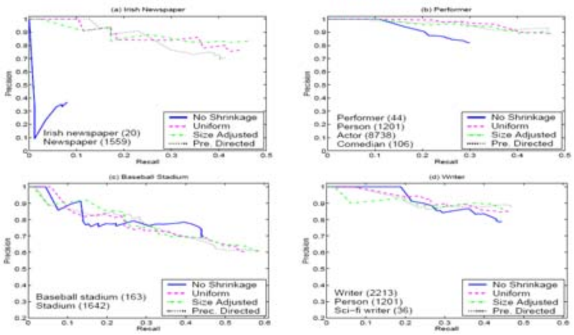
Improving Recall on Sparse Classes

[Wu et al. KDD-08]

- **Shrinkage**
  - Extra Training Examples from Related Classes
  - How Weight New Examples?



Improvement due to Shrinkage



Improving Recall on Sparse Classes

[Wu et al. KDD-08]

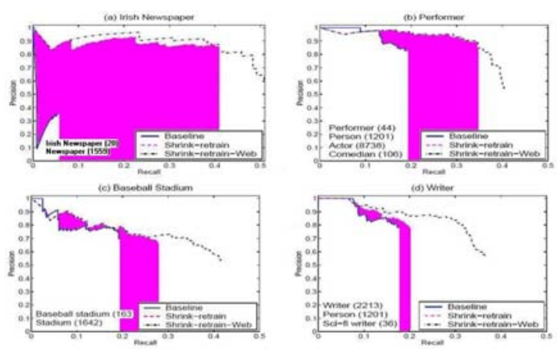
Retraining

- Compare Kylin Extractions with Tuples from Textrunner
- Additional Positive Examples
- Eliminate False Negatives

TextRunner [Banko et al. IJCAI-07, ACL-08]

- Relation-Independent Extraction
- Exploits Grammatical Structure
- CRF Extractor with POS Tag Features

Recall after Shrinkage / Retraining…

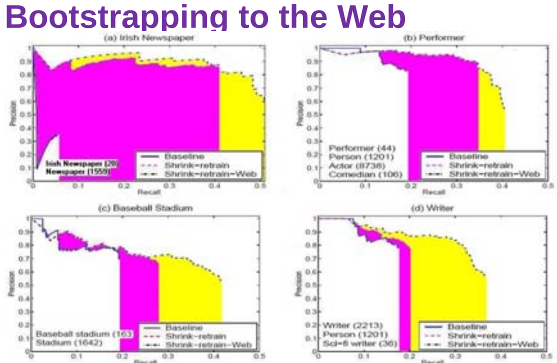


Improving Recall on Sparse Classes

[Wu et al. KDD-08]

- **Shrinkage**
- **Retraining**
- **Extract from Broader Web**
  - **44% of Wikipedia Pages = "stub"**
    - **Extractor quality irrelevant**
  - **Query Google & Extract**
    - **How maintain high precision?**
    - **Many Web pages noisy, describe multiple objects**
    - **How integrate with Wikipedia extractions?**

Bootstrapping to the Web



Main Lesson: Self Supervision

- **Find structured data source**
- **Use heuristics to generate training data**
  - E.g. Infobox attributes & matching sentences

Self-supervised Temporal Extraction

- **Goal Extract:**
  - happened(recognizes(UK, China), 1/6/1950)

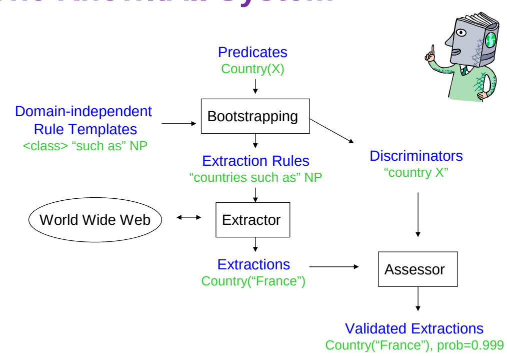
Other Sources

- **Google News Archives**

Methods for Open IE

- **Self Supervision**
  - Kylin (Wikipedia)
  - Shrinkage & Retraining
  - Temporal Extraction
- **Hearst Patterns**
  - PMI Validation
  - Subclass Extraction
- **Pattern Learning**
- **Structural Extraction**
  - List Extraction & WebTables
  - TextRunner

The KnowItAll System



Unary predicates: instances of a class

Unary predicates:
instanceOf(City), instanceOf(Film), instanceOf(Company), …

Good recall and precision from generic patterns:
<class> "such as" X
X "and other" <class>

Instantiated rules:
"cities such as" X X "and other cities"
"films such as" X X "and other films"
"companies such as" X X "and other companies"

Recall – Precision Tradeoff

High precision rules apply to only a small percentage of sentences on Web

	hits for "X"	"cities such as X"	"X and other cities"
Boston	365,000,000	15,600,000	12,000
Tukwila	1,300,000	73,000	44
Gjatsk	88	34	0
Hadaslav	51	1	0

"Redundancy-based extraction" ignores all but the unambiguous references.

Limited Recall with Binary Rules

Relatively high recall for unary rules:

"companies such as" X	2,800,000 Web hits
X "and other companies"	500,000 Web hits

Low recall for binary rules:

X "is the CEO of Microsoft"	160 Web hits
X "is the CEO of Wal-mart"	19 Web hits
X "is the CEO of Continental Grain"	0 Web hits
X ", CEO of Microsoft"	6,700 Web hits
X ", CEO of Wal-mart"	700 Web hits
X ", CEO of Continental Grain"	2 Web hits

Examples of Extraction Errors

Rule: countries such as X => instanceOf(Country, X)

"We have 31 offices in 15 countries such as London and France."

=> ~~instanceOf~~(Country, London)

instanceOf(Country, France)

Rule: X and other cities => instanceOf(City, X)

"A comparative breakdown of the cost of living in Klamath County and other cities follows."

=> ~~instanceOf~~(City, Klamath County)

"Generate and Test" Paradigm

1. Find extractions from generic rules
2. Validate each extraction
   - Assign probability that extraction is correct
   - Use search engine hit counts to compute PMI
   - PMI (pointwise mutual information) between
     - extraction
     - "discriminator" phrases for target concept

PMI-IR: P.D.Turney, "Mining the Web for synonyms: PMI-IR versus LSA on TOEFL". In Proceedings of ECML, 2001.

Computing PMI Scores

Measures mutual information between the extraction and target concept.

$$PMI(D,I) = \frac{|hits(D+I)|}{|hits(I)|}$$

I = an instance of a target concept
instanceOf(Country, "France")

D = a discriminator phrase for the concept
"ambassador to X"

D+I = insert instance into discriminator phrase
"ambassador to France"

Example of PMI

- Discriminator: "countries such as X"
- Instance: "France" vs. "London"
- PMI for France >> PMI for London (2 orders of mag.)
- Need features for probability update that distinguish
  - "high" PMI from "low" PMI for a discriminator

"countries such as France" : 27,800 hits
"France": 14,300,000 hits

$$PMI = \frac{27{,}800}{14{,}300{,}000} = 1.94E^{-3}$$

"countries such as London" : 71 hits
"London": 12,600,000 hits

$$PMI = \frac{71}{12{,}600{,}000} = 5.6E^{-6}$$

PMI for Binary Predicates

$$PMI(D,I_1,I_2) = \frac{|hits(D+I_1+I_2)|}{|hits(I_1,I_2)|}$$

$hits(D+I_1+I_2)$ insert both arguments of extraction into the discriminator phrase

$hits(I_1,I_2)$ each argument is a separate query term

Extraction: CeoOf("Jeff Bezos", "Amazon")
Discriminator: <arg1> ceo of <arg2>
PMI = 0.017

~~670~~ hits for "Jeff Bezos ceo of Amazon"
39,000 hits for "Jeff Bezos", "Amazon"

Bootstrap Training

1. Only input is set of predicates with class labels.
   instanceOf(Country), class labels "country", "nation"
2. Combine predicates with domain-independent templates
   <class> such as NP => instanceOf(class, NP)
   to create extraction rules and discriminator phrases
   rule: "countries such as" NP => instanceOf(Country, NP)
   discrim: "country X"
3. Use extraction rules to find set of candidate seeds
4. Select best seeds by average PMI score
5. Use seeds to train discriminators and select best discriminators
6. Use discriminators to rerank candidate seeds, select new seeds
7. Use new seeds to retrain discriminators, ….

Bootstrap Parameters

- Select candidate seeds with minimum support
  - Over 1,000 hit counts for the instance
  - Otherwise unreliable PMI scores
- Parameter settings:
  - 100 candidate seeds
  - Pick best 20 as seeds
  - Iteration 1, rank candidate seeds by average PMI
  - Iteration 2, use trained discriminators to rank candidate seeds
  - Select best 5 discriminators after training
    - Favor best ratio of $P(PMI > thresh | \phi)$ to $P(PMI > thresh | \neg\phi)$
    - Slight preference for higher thresholds
- Produced seeds without errors in all classes tested

Discriminator Phrases from Class Labels

From the class labels "country" and "nation"

country X	nation X
countries X	nations X
X country	X nation
X countries	X nations

Equivalent to weak extraction rules
- no syntactic analysis in search engine queries
- ignores punctuation between terms in phrase

PMI counts how often the weak rule fires on entire Web
- low hit count for random errors
- higher hit count for true positives

Discriminator Phrases from Rule Keywords

From extraction rules for instanceOf(Country)

countries such as X	nations such as X
such countries as X	such nations as X
countries including X	nations including X
countries especially X	nations especially X
X and other countries	X and other nations
X or other countries	X or other nations
X is a country	X is a nation
X is the country	X is the nation

Higher precision but lower coverage than discriminators from class labels

Using PMI to Compute Probability

Standard formula for Naïve Bayes probability update
- useful as a ranking function
- probabilities skewed towards 0.0 and 1.0

$$P(\phi | f_1, f_2, \ldots f_n) = \frac{P(\phi)\prod_i P(f_i | \phi)}{P(\phi)\prod_i P(f_i | \phi) + P(\neg\phi)\prod_i P(f_i | \neg\phi)}$$

Probability that fact ϕ is a correct, given features $f_1, f_2, \ldots f_n$

Need to turn PMI-scores into features $f_1, f_2, \ldots f_n$

Need to estimate conditional probabilities $P(f_i | \phi)$ and $P(f_i | \neg\phi)$

Features from PMI: Method #1

Thresholded PMI scores

Learn a PMI threshold from training

Learn conditional probabilities for PMI > threshold,

given that ϕ is in the target class, or not

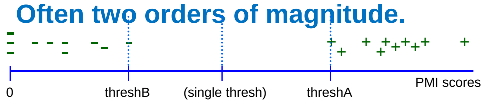
P(PMI > thresh | class) P(PMI <= thresh | class)

P(PMI > thresh | not class) P(PMI <= thresh | not class)

One Threshold or Two?

Wide gap between positive and negative training.

Often two orders of magnitude.



With two thresholds, learn conditional probabilities:

P(PMI > threshA | class)	P(PMI > threshA | not class)
P(PMI < threshB | class)	P(PMI < threshB | not class)
P(PMI between A,B | class)	P(PMI between A,B | not class)

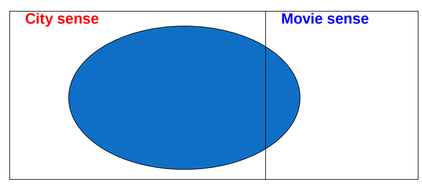
Polysemy

- **Problems with Polysemy**
  - Low PMI if instance has multiple word senses
  - False negative if target concept is not the dominant word sense.
- **"Amazon" as an instance of River**
  - Most references are to the company, not the river
- **"Shaft" as an instance of Film**
  - 2,000,000 Web hits for the term "shaft"
  - Only a tiny fraction are about the movie

Chicago

$$PMI\ (I, D, C) = \frac{|\ Hits\ (I + D \mid C)\ |}{|\ Hits\ (I \mid C)\ |}$$

Chicago Unmasked



$$\frac{|\ Hits\ (Chicago\ + Movie\ - City\ )\ |}{|\ Hits\ (Chicago\ - City\ )\ |}$$

Impact of Unmasking on PMI

Name	Recessive	Original	Unmask	Boost
Washington	city	0.50	0.99	96%
Casablanca	city	0.41	0.93	127%
Chevy Chase	actor	0.09	0.58	512%
Chicago	movie	0.02	0.21	972%

Methods for Open IE

- **Self Supervision**
  - Kylin (Wikipedia)
  - Shrinkage & Retraining
  - Temporal Extraction
- **Hearst Patterns**
  - PMI Validation
  - Subclass Extraction
- **Pattern Learning**
- **Structural Extraction**
  - List Extraction & WebTables
  - TextRunner

How to Increase Recall?

RL: learn class-specific patterns.

“Headquarted in <city>”

SE: Recursively extract subclasses.

“Scientists such as physicists and chemists”

LE: extract lists of items

(~ Google Sets).

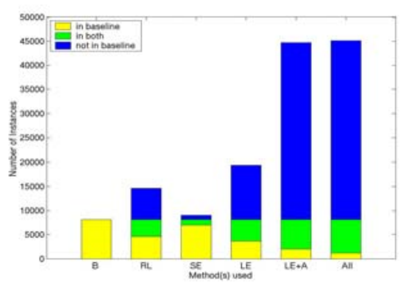
List Extraction (LE)

1. **Query Engine with known items.**
2. **Learn a wrapper for each result page.**
3. **Collect large number of lists.**
4. **Sort items by number of list “votes”.**

LE+A**=sort list according to Assessor.**

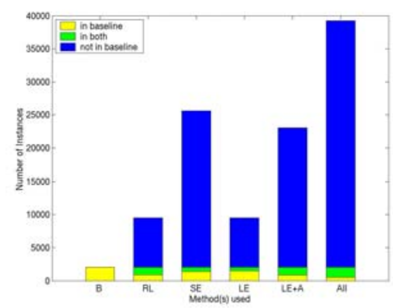
Evaluation: Web recall, at precision= 0.9.

Results for City



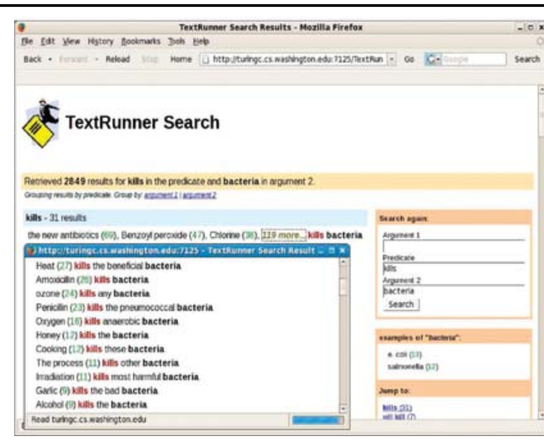
Found 10,300 cities missing from Tipster Gazetteer.

Results for Scientist



Methods for Open IE

- **Self Supervision**
  - Kylin (Wikipedia)
  - Shrinkage & Retraining
  - Temporal Extraction
- **Hearst Patterns**
  - PMI Validation
  - Subclass Extraction
- **Pattern Learning**
- **Structural Extraction**
  - List Extraction & WebTables
  - TextRunner



TextRunner

Relative Frequency	Category	Simplified Lexico-Syntactic Pattern
37.8	Verb	E_1 Verb E_2 *X established Y*
22.8	Noun + Prep	E_1 NP Prep E_2 *X settlement with Y*
16.0	Verb + Prep	E_1 Verb Prep E_2 *X moved to Y*
9.4	Infinitive	E_1 to Verb E_2 *X plans to acquire Y*
5.2	Modifier	E_1 Verb E_2 Noun *X is Y winner*
1.8	Coordinate_n	E_1 (and|,|-|:) E_2 NP *X-Y deal*
1.0	Coordinate_v	E_1 (and|,) E_2 Verb *X , Y merge*
0.8	Appositive	E_1 NP (:|,)? E_2 *X hometown : Y*

The End

(Formerly) Open Question #3

- **Sparse data (even with entire Web)**
  - PMI thresholds are typically small (1/10,000)
  - False negatives for instances with low hit count
- **City of Duvall**
  - 312,000 Web hits
  - Under threshold on 4 out of 5 discriminators
- **City of Mossul**
  - 9,020 Web hits

See next talk…

Future Work III
Creating Better CRF Features?

training data

A.B. is a Swedish politician.
Y.P. was a Russian numismatist.
J.F.S.d.L. is a Brazilian attacking midfielder.
B.C.A. is a Danish former football player.

now:

1 isCapitalized
0 contains n-gram "ed"
…
0 equals "Swedish"
1 equals "Russian"
0 equals "politician"
0 equals "B.C.A."

test data

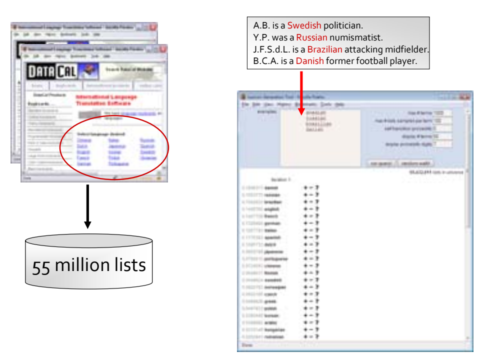
N.F. was an Australian.
R.S. is a Fijian politician.
A. I. is a Boeing subsidiary.
D.F. is a Swedish actor.

would like:

0 is on list of occupations
1 is on list of nationalities
is on list of first names
is on list of …
0
0
0
0

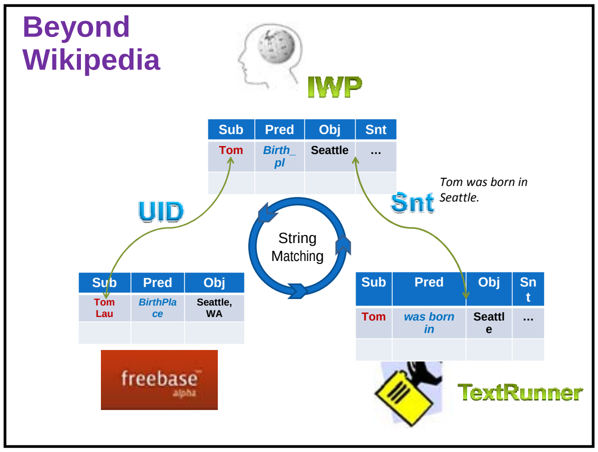
But where do we get the lists from?

Mining Lists from the Web



Seeds From CRFs

Unified List


Beyond Wikipedia
IWP
Sub Pred Obj Snt
Tom Birth_pl Seattle …
UID
String Matching
Snt
Tom was born in Seattle.
Sub Pred Obj
Tom Lau BirthPlace Seattle, WA
Sub Pred Obj Snt
Tom was born in Seattle …
freebase alpha
TextRunner