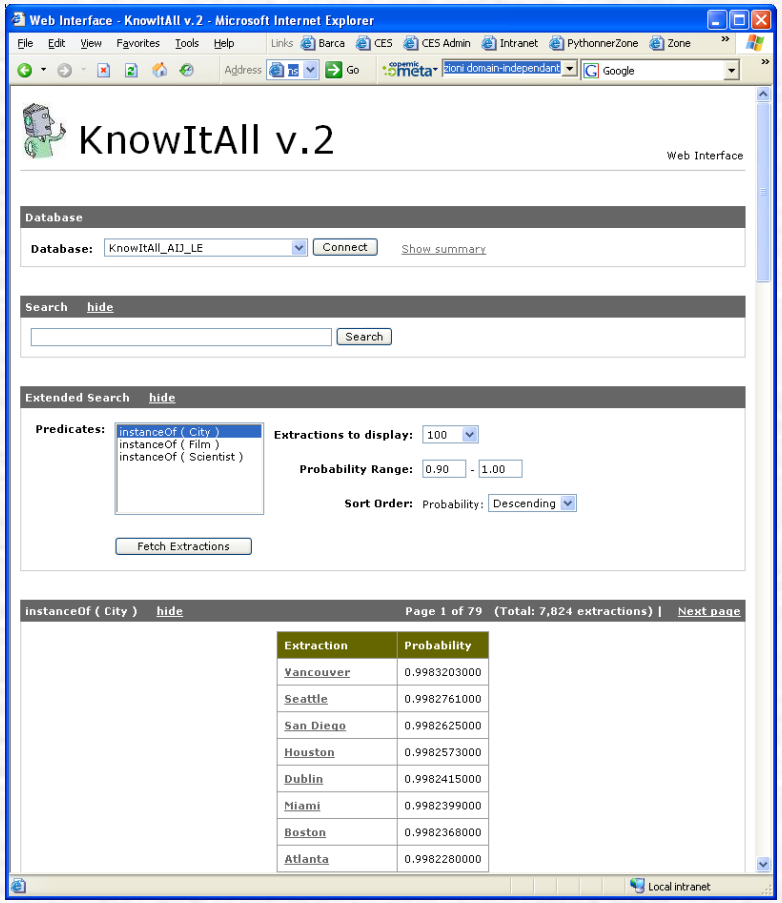
IE Systems

Contents of the Chapter

- IE Systems:
  - KnowItAll System

KnowItAll System

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



Publication:

" methods for Domain-independent information extraction from the Web:an experimental comparison" (aaai2004)

Dept.of CS, University of Washington

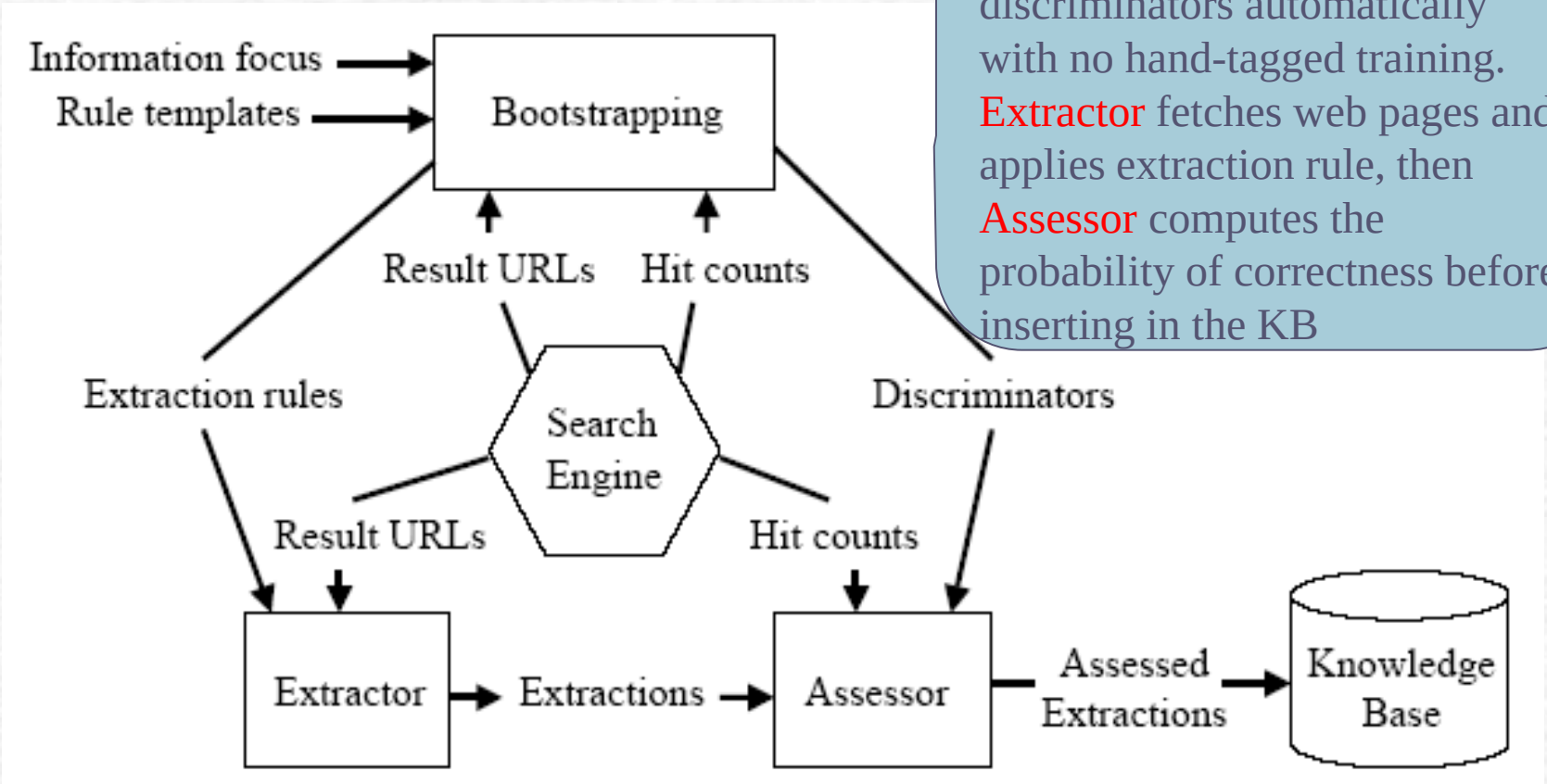
KnowItAll System (overview)

- Autonomous, domain-independant system that extract **facts**, **concepts**, and **relationships** from the Web.

1. Focus (e.g.: city)
2. Patterns instanciation:
   NP1 such as *NP2* = « city such as »
   Plural(NP1) such as NP2-List = « cities such as »
3. Search + passage retrieval:
   … a *city such as* **Sudbury**, at north of the Great Lakes…
   …*cities such as* **Chicago**, **New York**, **Atlanta** and **Orlando** …
4. Assessor: PMI-IR → Hits(Atlanta AND city) / Hits (Atlanta)

Flowchart of main components in KnowItAll

Bootstrapping creates extractions rules and discriminators automatically with no hand-tagged training. Extractor fetches web pages and applies extraction rule, then Assessor computes the probability of correctness before inserting in the KB



Bootstrapping input:

Information Focus, a set of predicates

Predicate: City
labels: "city", "town"

Predicate: Country
labels: "country", "nation"

Predicate: capitalOf(City,Country)
relation labels: "capital of"
class-1 labels: "city", "town"
class-2 labels: "country", "nation"

Predicate: Film
labels: "film", "movie"

Predicate: MovieActor
labels: "actor", "movie star"

Predicate: starsIn(MovieActor,Film)
relation labels: "stars in", "star of"
class-1 labels: "actor", "movie star"
class-2 labels: "film", "movie"

Bootstrapping:
From Rule Template to Extraction Rules

Predicate:	Class1
Pattern:	NP1 "such as" NPList2
Constraints:	head(NP1)= plural(label(Class1)) & properNoun(head(each(NPList2)))
Bindings:	Class1(head(each(NPList2)))

Rule Template

Predicate:	City
Pattern:	NP1 "such as" NPList2
Constraints:	head(NP1)= "cities" properNoun(head(each(NPList2)))
Bindings:	City(head(each(NPList2)))
Keywords:	"cities such as"

An extraction rule

Predicate City produces:

cities such as NPList and towns such as NPList

Extractions based on rules

Examples:

- The tour includes major cities such as New York, central Los Angeles and Dallas. →New York, Dallas are cities
- Detailed maps and information for several cities such as airport maps, city and downtown maps
- → **use discriminator to assess**

Main Components of KnowItAll

- The Extractor:
- matches rules to Web pages returned for the query.
- Creates one or more extractions
- The Assessor:
- Discriminator: to check whether the extractions are correct or not.
- Na ïve Bayes Classifier: the mutual information statistics are treated as features that input to the classifier

This is the probability that fact φ is correct, given features f1,f2,..

What is discriminator?

- **Discriminator:** an extraction pattern with alternating context strings and slots.
- **How to evaluate?** D: a discriminator, I is a result

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

- Example: ‘New York’ or ‘airport map’ are cities?

Discriminators for class City:

<I> is a city→ query: <New York> is a city→ return many pages

query: <airport map> is a city→ return few pages

Discriminator (cont.)

- Each discriminator has a threshold.
- If the PMI of a discriminator has greater than the threshold, the result is true.

The discriminator for class city are:

Discriminator: <I> is a city
Learned Threshold T: 0.000016
P(PMI > T | class) = 0.83
P(PMI > T | ¬class)= 0.08

Discriminator: cities such as <I>
Learned Threshold T: 0.0000053
P(PMI > T | class) = 0.75
P(PMI > T | ¬class)= 0.08

Discriminator: <I> and other towns
Learned Threshold T: 0.00000075
P(PMI > T | class) = 0.83
P(PMI > T | ¬class)= 0.08

Discriminator: cities including <I>
Learned Threshold T: 0.0000047
P(PMI > T | class) = 0.75
P(PMI > T | ¬class)= 0.08

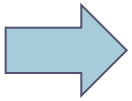
Discriminator: cities <I>
Learned Threshold T: 0.00044
P(PMI > T | class) = 0.91
P(PMI > T | ¬class)= 0.25

Discriminator (cont.)

- Extraction result : Fes is a city.
- Discriminator: "Fes" is a city"

- Search engine returns: "Fes" has 446,000 hits," "Fes is a city" has 14 hits. PMI scoring=0.000031

Discriminator: <I> is a city
Learned Threshold T: 0.000016
P(PMI > T | class) = 0.83
P(PMI > T | ¬class)= 0.08



Fes is a city

Discussion

How to implement it?

High-level pseudocode for KowItAll

KNOWITALL(information focus I, rule templates T)
    Bootstrap(I, T) sets rules R, queries Q, and discriminators D
    Do until queries in Q are exhausted (or other termination criterion)
        Extractor(R, Q) writes extractions list E
        Assessor(E, D) adds extractions to the knowledgebase

Extractor(rules R, queries Q)
    Select queries from Q, set the number of downloads for each query
    Send selected queries to search engines
    For each webpage w whose URL was returned by a search engine
        Extract fact e from w using the rule associated with the query
        Write e to extractions list E

Assessor(extraction list E, discriminators D)
    For each extraction e in E
        Assign a probability p to e using a Bayesian classifier based on D
        Add e,p to the knowledgebase

BOOTSTRAP(information focus I, rule templates T)

R = generate rules from T for each predicate in I

Q = generate queries associated with each rule in R

U = generate untrained discriminators from rules in R, class names in I

Use Q to find at least n candidate seeds for each predicate in I
with hit counts $> h$

First Iteration:

S = select m candidate seeds for each predicate in I
with highest average PMI over U

D = train U on S, select best k discriminators for each predicate in I

Subsequent Iterations:

S = select m candidate seeds for each predicate in I
with highest probability from D

D = train U on S, select best k discriminators for each predicate in I

Figure 10: Pseudocode for Bootstrapping.

KnowItAll System

- Given focus, predefined rule template
- Extract many facts , assess them and put into Database.
- → develop more extended functions in order to increase precision and recall

Three extended Functions

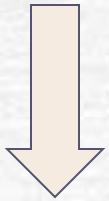
- Rule→Pattern Learning (PL): learns domain-specific patterns as extraction rules and validation patterns.
- Subclass Extraction (SE): automatically identifies subclasses and instances of these subclasses. E.g. physicists are scientists.
- List Extraction (LE): locates lists of class instances, learns a wrapper for each list and uses wrapper to extract list elements

Rule Learning (RL):

learns domain-specific rules and validates the accuracy of instances they extract.

Patterns

"city, such as **Boston**"
"mega-city such as **Mexico**"
"within a city, such as **Rice University**"



Facts

(with likelihood)

PMI(Boston, city) = 0,0000060
PMI(Mexico, city) = 0,0000056
PMI(Rice University, city) = 0,0000024

Rule → Pattern Learning (PL)

Facts

(most probable)

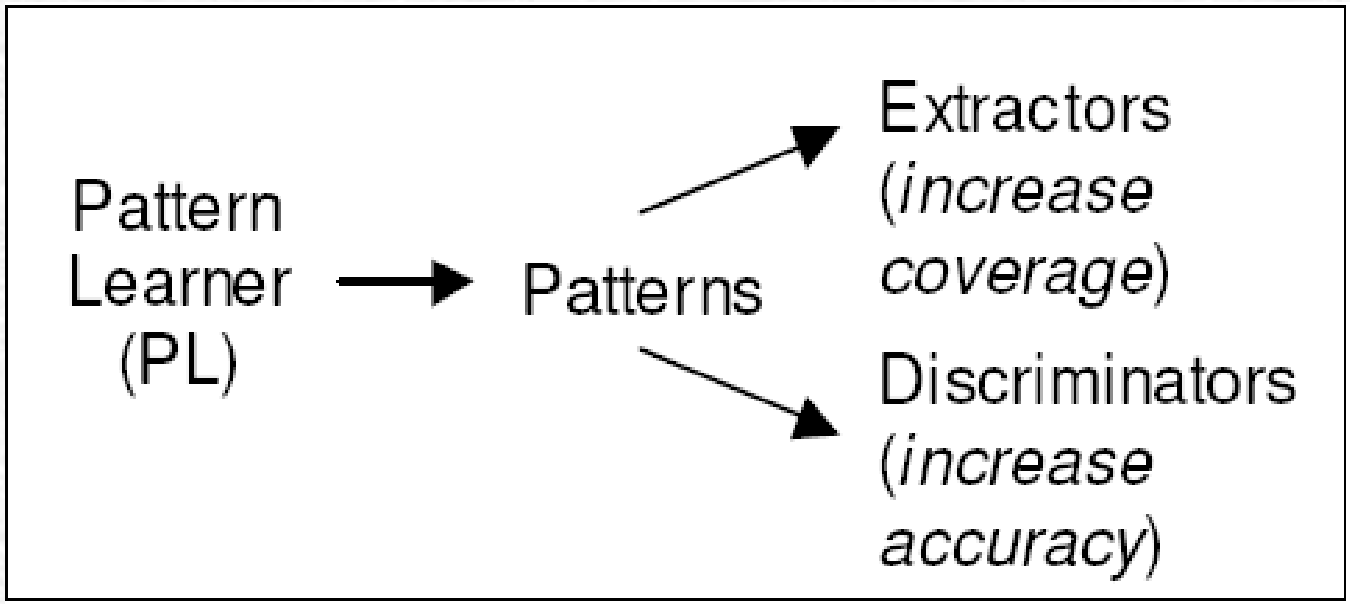
of **Boston** College
the **Boston** Globe
a **Boston** Parking Space
headquartered in Boston
Crime in **Mexico** continues
Mexico City Hotels
headquartered in Mexico

New patterns

Headquartered in *NP*

Pattern Learning

- Aim: increase both coverage and accuracy



Pattern Learning (Process)

1. Start with set I of seed instances generated by domain-independent extractors
2. For each seed instance i in I: issue a query to web search engine for i and for each occurrence of i in the returned documents record a context string comprised of w words before i, a placeholder and the w words after i
3. Output the best patterns according to some metric.

Pattern Learning (PL)

- Estimating pattern quality by *recall* and *precision*.

$$EstimatedPrecision = \frac{c(p) + k}{c(p) + n(p) + m}$$

$$EstimatedRecall = \frac{c(p)}{S}$$

- C(p): number of distinct seeds from the target class.
- N(p:):distinct seeds from other classes. The positive examples of one class are the negative examples for all other classes.
- S is the total number of seeds in the target class, k/m is a constant prior estimate of precision using PMI assessment.

Rules→Patterns examples

Three of the most productive rules for each class, along with the number of correct extractions produced by each rule and the rule's overall precision before assessment.

Rule	*Correct Extractions*	*Precision*
the cities of <*city*>	5215	0.80
headquartered in <*city*>	4837	0.79
for the city of <*city*>	3138	0.79
in the movie <*film*>	1841	0.61
<*film*> the movie starring	957	0.64
movie review of <*film*>	860	0.64
and physicist <*scientist*>	89	0.61
physicist <*scientist*>,	87	0.59
<*scientist*>, a British scientist	77	0.65

Learned Patterns as extractors

- Select top patterns :

✓ Prefer patterns that *appear for multiple distinct seeds*.

✓ Sort the remaining patterns according to their estimated precision.

Learned Patterns as Discriminators

- Better than domain independent discriminators for the classifier.
- Better on some extractions than it does on others.

  e.g. "cities such as <city>" has high precision, but appears only rarely on the Web.

  "<city> hotels" might offer better performance.

Subclass Extraction (SE)

Automatically identifies subclasses in order to facilitate extraction.

- Goal: increase the recall of *KnowItAll*

  Focus: scientist

  Pattern: «scientist such as NP»

  … scientist such as **Arthur Noyes**

  … scientist such as **Isaac Newton**

  … scientist such as **Sandra Steingraber**

Subclass Extraction (SE)

- Using found facts, apply the reverse pattern:

  « N such as Arthur Noyes »

  « **chemist** such as Arthur Noyes »

  « **biologist** such as Sandra Steingraber

Chemist is a scientist

Biologist is a scientist

Some Patterns for SE

Pattern	Extraction
$C_1 \{ \text{“,”} \}$ “such as” CN	$isA(CN, C_1)$
“such” C_1 “as” CN	$isA(CN, C_1)$
$CN \{ \text{“,”} \}$ “and other” C_1	$isA(CN, C_1)$
$CN \{ \text{“,”} \}$ “or other” C_1	$isA(CN, C_1)$
$C_1 \{ \text{“,”} \}$ “including” CN	$isA(CN, C_1)$
$C_1 \{ \text{“,”} \}$ “especially” CN	$isA(CN, C_1)$
C_1 “and” CN	$isA(CN, class(C_1))$
$C_1 \{ \text{“,”} \} C_2 \{ \text{“,”} \}$ “and” CN	$isA(CN, class(C_1))$

Context-independent and context-dependent subclasses

- Context: refers to a set of keywords provided by the user that suggest a knowledge domain of interest.

- For example: Subclasses of Person in a **Pharmaceutical context**, Priest (n.牧师) is probably not a good candidate where pharmacist(药剂师)is.

Named subclasses and derived subclasses

- Named subclasses are represented by novel terms, such as: *Capital* is a named subclass of *City.*
- Derived subclasses are phrases whose head noun is the same as the name of the super class, such as: *European city* is a derived sub-class of *City.*

SE Assessing

- **Morphology checking**: e.g.: microbiologist is a subclass of "biologist".
- **WordNet checking**: whether a subclass is a hyponym of the class in WordNet, if so, it assigns it a very high probability.
- **PMI-test**: computes co-occurrence statistics of candidate terms with a set of class discriminators.

Sample Result of SE

biologist	zoologist
astronomer	meteorologist
mathematician	economist
geologist	sociologist
chemist	oceanographer
anthropologist	pharmacist
psychologist	climatologist
paleontologist	neuropsychologist
engineer	microbiologist

Table 3: **Subclasses of Scientist found by SE.**

SE experiments

- Context-independent classes (scientist and Film)

Method	**Scientist**				**Film**			
	Precision	Recall	NW	Total	Precision	Recall	NW	Total
SE_{base}	0.91	0.28	0.08	11	1.0	0.36	0.5	8
SE_{self}	0.87	0.69	0.15	27	0.94	0.77	0.82	**17**
SE_{iter}	0.84	0.74	0.17	**29**	0.93	0.68	0.8	16

Total : the number of correct sub-classes found.

NW: the proportion of correct subclasses missing from WordNet.

SE_{self} Se_{iter} are two extensions in order to increase its recall.

SE experiments

- Three Pharmaceutical domain classes: People, Product, Organization
- Simple features are such good predictors of a candidate representing a subclass.
- Extracted subclasses have overlap with the hyponym information in WordNet. E.g. supplier, exporter etc.

Method	People				Organization				Product			
	Precision	Recall	NW	Total	Precision	Recall	NW	Total	Precision	Recall	NW	Tota
SE_{base}	1.0	0.28	0.07	14	0.92	0.20	0.09	11	0.88	0.44	1.0	31
SE_{self}	1.0	0.86	0.02	42	0.87	0.84	0.36	47	0.86	0.74	1.0	51
SE_{iter}	0.95	0. 94	0.02	**46**	0.89	0.95	0.22	**52**	0.84	0.88	1.0	**62**

List Extraction (LE)

Locates lists of class instances, learns a "wrapper" for each list, and uses the wrapper to extract list elements

- Goal: increase the recall of *KnowItAll*

Find web pages with set (k=4) of random facts.

query : « **chicago** AND **boston** AND **mexico** AND **buenos aires** »

repeat 5,000-10,000 times

In each document, try to find « a list »

List Extraction (LE)

List Extraction (LE)

- parses the document and converts to a DOM tree
- selects subtrees containing positive examples
- computes the greatest common prefixes and suffixes of these examples
- chooses header and tail strings to limit extraction to good subtree.

LE Algorithm

Input: a set of positive examples

Output: a list of tokens (extractions)

```
LISTEXTRACTOR(seedExamples)
    documents = searchForDocuments(seedExamples)
    For each document in documents
        parseTree = ParseHTML(document)
        For each subtree in parseTree
            keyWords = findAllSeedsInTree(subtree)
            prefix = findBestPrefix(keyWords, subtree)
            suffix = findBestSuffix(keyWords, subtree)
            Add to wrapperTree from createWrapper(prefix, suffix))
        For each goodWrapper in wrapperTree
            Find extractions using goodWrapper
    Return list of extractions
```

An example:

```
Keywords: Italy, Japan, Spain, Brazil
```

```
1    <html>
2      <body>
3        My favorite countries:
4        <table>
5          <tr><td><a>Italy</a></td><td><a>Japan</a></td><td><a>France</a></td></tr>
6          <tr><td><a>Israel</a></td><td><a>Spain</a></td><td><a>Brazil</a></td></tr>
7        </table>
8        My favorite pets:
9        <table>
10         <tr><td><a>Dog</a></td><td><a>Cat</a></td><td><a>Alligator</a></td></tr>
11       </table>
12     </body>
13   </html>

Wrappers (at least 2 keywords match):
   w1 (1 - 13):  <td><a>TOKEN</a></td>
   w2 (2 - 12):  <td><a>TOKEN</a></td>
   w3 (4 - 7):   <td><a>TOKEN</a></td>
   w4 (5 - 5):   <td><a>TOKEN</a></td><td><a>
   W5 (6 - 6):   </a></td><td><a>TOKEN</a></td>
```

w1,w2: too general

w3: the best wrapper

w4, w5 : overfitting the suffix is too long

An example (cont.)

Parameters to be considered when using the algorithm:

- Number of keywords to match in a block
- Selection of wrappers from the wrapper tree
- Length/complexity of prefix/suffix/both
- Number of search words to use for retrieving documents
- Selection of keywords for searching

A sample of LE result

Class	Seeds	Queries	Extractions	Correct	% Correct
City	3,000	9,000	190,000	90,000	47%
Film	300	9,000	31,000	24,500	79%
Scientist	50	5,000	65,000	15,000	23%
City	5	1	6,000	4,000	66%

Experiment 8: Results for LE. *Seeds* is the number of positive examples given as input. *Queries* is the number of times 4 tokens were randomly selected from the seeds to search for documents. *Extractions* is the total number of unique extractions. LE can find large numbers of extractions from relatively few queries. *Correct* is the number of extractions in the class before using the Assessor to boost precision.

The method (B) and three extended functions (PL,SE,LE)

Class	Method	Extractions	Correct	Precision	Corr. at Precision .90	Corr. at Precision .80
City	B	10,094	8,342	0.83	5,852	8,342
City	PL	11,338	7,442	0.66	5,883	6,548
City	SE	5,045	3,514	0.70	2,023	2,965
City	LE	151,016	78,157	0.52	20,678	33,136
Film	B	36,739	21,859	0.59	4,645	7,436
Film	PL	15,306	9,755	0.64	2,286	2,648
Film	SE	16,820	9,840	0.57	2,286	4,424
Film	LE	78,859	61,418	0.72	27,973	55,575
Scientist	B	4,781	3,690	0.77	1,599	2,905
Scientist	PL	4,519	2,119	0.47	751	1,869
Scientist	SE	6,807	6,168	0.91	6,168	6,168
Scientist	LE	15,907	10,147	0.64	1,245	3,773

Table 4: The total number of extractions, total number correct, and overall precision for each class and method. The total number of correct extractions greatly exceeds the number of correct extractions at precision .80, which suggests that our current Assessor achieves high precision at the cost of a large number of false negatives.

Summarize for KnowItAll

- KnowItAll is an domain-independent Information extraction system
- The only input is a «**focus**» (city, scientist, movie, …) **Pattern instanciation**, **passage retrieval, PMI-IR test**.
- Bootstrap learning (obviates hand-labeled training examples)
- Reling on the scale and redundancy of the Web
- Extracting n-ary relations from the Web
- RL, SE and LE improve extraction recall
- **Over half of correct instances of City were not in the Tipster Gazetteer**.

Classroom Discussion

- From your point, what are the advantages and disadvantages of the KnowItAll system?
- What are the potential challenges?

References

- Oren Etzioni, et al.”Unsupervised named-entity extraction from the Web: an experimental study” in the proceeding of AAAI ’04 paper.
- Michele Banko, et al.” Open information extraction from the Web” in proceeding of IJCAJ 07.