KnowItAll

Oren Etzioni, Stephen Soderland , Daniel Weld
Michele Banko, Alex Beynenson, Jeff Bigham, Michael Cafarella, Doug Downey, Dave Ko, Stanley Kok, Jim Li, Mike Lindemark, Tessa McDuff, Bao Nguyen, Ana-Maria Popescu, Stefan Schoenmackers, Tal Shaked, Junji Tomita, and Alex Yates

Ongoing work since 2003

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

Goals of KnowItAll

Information extraction from the Web that is:

- Domain-independent
- Genre-independent
- Unsupervised
- Massively scalable
- High precision
- Fuses information across documents

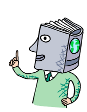
KnowItAll Projects

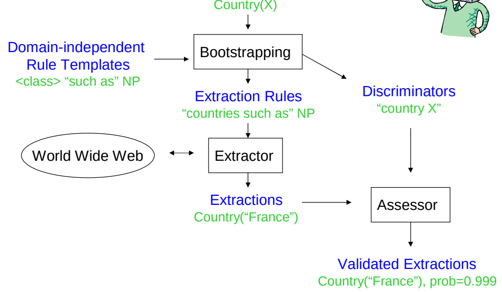
KnowItAll (baseline system)
  Unsupervised information extraction
  Google queries for extraction and verification
KnowItNow (massive speedup)
  BE: novel search engine index
  Urns: formal probability model for verification
Opine (mining on-line reviews)
  Learn attributes of a product
  Find strength and orientation of opinions
TextRunner
  Semantic graph of relationships from corpus
  Question-answering based on relation graph
Ontology Learning
  Learn relations and attributes of arbitrary classes
  Continuously grow from a small knowledge base

Unsupervised Information Extraction

1. Create extraction rules from generic rule templates
2. Send queries to search engine based on extraction rules
3. Extract information from resulting pages
4. Verify extractions with mutual information from Web hitcounts (PMI-IR)
5. Enter extractions into a database

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"

Binary Predicates

Domain-independent rule templates:

relation(arg1, arg2)	<arg1> “,” <relation> “of” <arg2>

Instantiated rules before binding an argument:

CeoOf(Person, Company)	<person> “, CEO of” <company>
StarsIn(Actor, Film)	<actor> “, star of” <film>
Population(Number, Country)	<number> “, population of” <country>

After binding an argument to an entry in knowledge base:

CeoOf(Person, Company)	NP “, CEO of WalMart”
StarsIn(Actor, Film)	Dustin Hoffman “, star of” NP
Population(Number, Country)	<number> “, population of France”

Instantiating a Rule Template

Rule Template (domain-independent)
- Predicate: predName(Class1)
- Pattern: NP1 “such as” NPList2
- Contraints: head(NP1) = plural(label(Class1)
  properNoun(head(each(NPList2)))
- Bindings: instanceOf(Class1, head(each(NPList2)))

Extraction Rule (substituting “instanceOf” and “Country”)
- Predicate: instanceOf(Country)
- Pattern: NP1 “such as” NPList2
- Contraints: head(NP1) = “nations”
  properNoun(head(each(NPList2)))
- Bindings: instanceOf(Country, head(each(NPList2)))
- Keywords: “nations such as”

Applying the Rule

Extraction Rule
- Predicate: instanceOf(Country)
- Pattern: NP1 “such as” NPList2
- Contraints: head(NP1) = “nations”
  properNoun(head(each(NPList2)))
- Bindings: instanceOf(Country, head(each(NPList2)))
- Keywords: “nations such as”

Sentence:
Other nations such as France, India and Pakistan, have conducted recent tests.

Three extractions:
- instanceOf(Country, France)
- instanceOf(Country, India)
- instanceOf(Country, Pakistan)

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

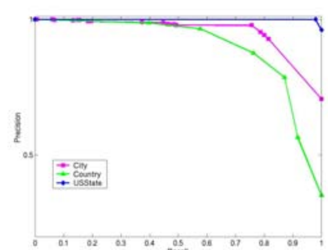
Relatively high recall for unary rules:

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

Low recall for binary rules:

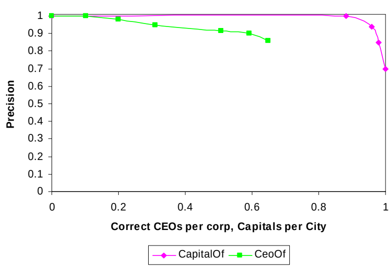
X “is the CEO of Microsoft”	160 Web hits
X “is the CEO of Wal-mart”	19 Web hits
X “is the CEO of Continental Grain”	0 Web hits
X “, CEO of Microsoft”	6,700 Web hits
X “, CEO of Wal-mart”	700 Web hits
X “, CEO of Continental Grain”	2 Web hits

Results for Unary Predicates



High precision and high recall for unary (instance of) extraction.
More errors for Country (“Latin America”, “Iriquois nation”, etc).

Results for Binary Predicates



High precision for both CapitalOf and CeoOf.
Able to find capitals of all cities.
Generic rules are too specific to find Ceo's for low-frequency companies.

"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.

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~~)

Computing PMI Scores

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

Measures mutual information between the extraction and target concept.

D = a discriminator phrase for the concept
"countries such as X"

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

D+I = insert the instance into discriminator phrase
"countries such as France"

Example of PMI

- Discriminator: "countries such as X"
- Instance: "France" vs. "London"
- PMI for France >> PMI for London (2 orders of magnitude)
- 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

$$\frac{\text{670 hits for "Jeff Bezos ceo of Amazon"}}{\text{39,000 hits for "Jeff Bezos", "Amazon"}}$$

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 $P(PMI > thresh \mid \phi)$ to $P(PMI > thresh \mid \neg\phi)$
    - 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

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

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 $P(f_i \mid \phi)$ and $P(f_i \mid \neg\phi)$

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

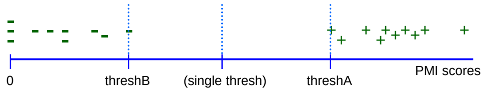
P(PMI > thresh | class)	P(PMI <= thresh | class)
P(PMI > thresh | not class)	P(PMI <= thresh | not class)

Small training set.

Train each discriminator separately.

One Threshold or Two?

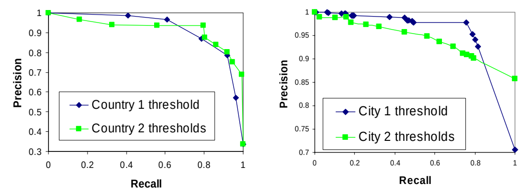
Wide gap between positive and negative training.
Often two orders of magnitude.



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)

Results for 1 or 2 thresholds



One threshold is better at high-precision end of curve.
Two thresholds are better at high-recall end of curve.

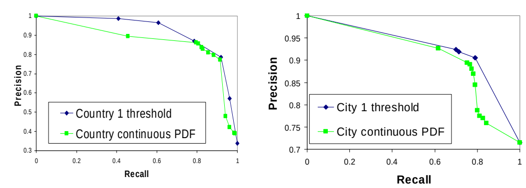
Features from PMI: Method #2

- Continuous Probability Density Function (PDF)
  - Gaussian probability model for positive PMI scores
  - Gaussian model for negative training.
  - Probability from Naïve Bayes is determined by ratio of P(PMI | class) to P(PMI | not class)

$$PDF_{f_i,\phi}(x) = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{(x-x_j)^2}{2\sigma^2}}$$

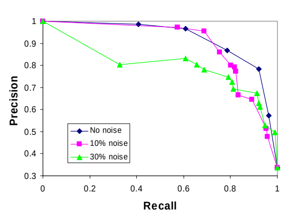
Where N is the number of positive training examples, x_j for discriminator f_j

PDF vs. Single Threshold



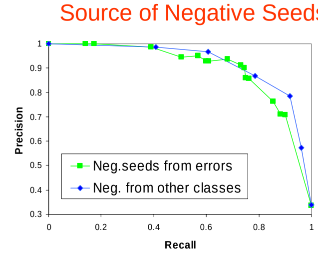
- Continuous PDF not as good as thresholded PMI
- Poor performance at both ends of precision-recall curve
  - Hard to smooth the small training size
  - Overfit to a few high PMI negative, low PMI positive training
  - Need to learn the ratio of $P(PMI \mid \phi)$ to $P(PMI \mid \neg\phi)$

Effect of Noisy Seeds



Bootstrapping must produce correct seeds
- 10% noise: some drop in performance
- 30% noise: badly degraded performance

Source of Negative Seeds



- Use seeds from other classes as negative training
  - Standard method for bootstrapping
- Better performance than negative training from hand-tagging extraction errors

Open Question #1

- 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
  - Under threshold on all 5 discriminators
  - PMI = 0.0 for 3 discriminators

Open Question #2

- 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

(Former) Open Question

- Time bottleneck
  - Search engine "courtesy wait" limits queries per day
  - Each instance requires several queries for PMI
- Hitcount caching helps with repeated experiments

Solved with KnowItNow
- BE enfolds extraction rules into search index
- Urns computes probabilities without hitcounts
- Speed up of 2 or 3 orders of magnitude

(Former) Open Question

- How to compute realistic probabilities
- Naïve Bayes formula gives skewed probabilities
  - Close to 1.0 probability or close to 0.0
  - Useful as ranking but not good probability estimates

Urns gives probabilities 15 times more accurate than PMI