Many thanks for lots of slides from many people, including Dan Jurafsky, Jim Martin, Oren Etzioni, Michele Banko, Rion Snow, Mike Mintz, and Steven Bills
Reminder!

• Lit Review paper due in 2 weeks!
• Start forming your project groups!
  • Working alone is fine, of course
  • But collaborating can be more fun!
• Bring your project ideas to office hours
• The ACL Anthology Searchbench may help find relevant literature: [http://aclasb.dfki.de/](http://aclasb.dfki.de/)
Extracting structured knowledge

Each article can contain hundreds or thousands of items of knowledge...

“The Lawrence Livermore National Laboratory (LLNL) in Livermore, California is a scientific research laboratory founded by the University of California in 1952.”
Goal: machine-readable summaries

Textual abstract:
Summary for human

Structured knowledge extraction:
Summary for machine
5 easy methods for relation extraction

1. Hand-built patterns
2. Supervised methods
3. Bootstrapping (seed) methods
4. Unsupervised methods
5. Distant supervision
5 easy methods for relation extraction

1. Hand-built patterns
2. Supervised methods
3. Bootstrapping (seed) methods
4. Unsupervised methods
5. Distant supervision
Bootstrapping approaches

• If you don’t have enough annotated text to train on...

• But you do have:
  • some seed instances of the relation
  • (or some patterns that work pretty well)
  • and lots & lots of unannotated text (e.g., the web)

• ... can you use those seeds to do something useful?

• Bootstrapping can be considered semi-supervised
Bootstrapping example

• Target relation: burial place

• Seed tuple: <Mark Twain, Elmira>

• Grep (Google) for “Mark Twain” and “Elmira”
  • “Mark Twain is buried in Elmira, NY.”
    → X is buried in Y
  • “The grave of Mark Twain is in Elmira”
    → The grave of X is in Y
  • “Elmira is Mark Twain’s final resting place”
    → Y is X’s final resting place

• Use those patterns to grep for new tuples

Slide adapted from Jim Martin
Bootstrapping à la Hearst

• Choose lexical relation $R$, e.g. hypernymy

• Gather a set of pairs that have this relation

• Find places in the corpus where these expressions occur near each other and record the environment

• Find the commonalities among these environments and hypothesize that common ones yield patterns that indicate the relation of interest

- Shakespeare and other authors
- metals such as tin and lead
- such diseases as malaria
- regulators including the SEC

- $X$ and other $Y$s
- $Y$s such as $X$
- such $Y$s as $X$
- $Y$s including $X$
Bootstrapping relations

Pattern-Based Relation Extraction

Tuple Search → Pattern Extraction

Pattern Extraction → Tuple Set

Tuple Set → Seed Patterns

Seed Patterns → Pattern Set

Pattern Set → Tuple Extraction

Tuple Extraction → Pattern Search

Pattern Search → Relational Table

Relational Table → Seed Tuples

Seed Tuples → Tuple Set

Slide adapted from Jim Martin
DIPRE (Brin 1998)

- Extract `<author, book>` pairs
- Start with these 5 seeds

<table>
<thead>
<tr>
<th>Author</th>
<th>Book</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isaac Asimov</td>
<td>The Robots of Dawn</td>
</tr>
<tr>
<td>David Brin</td>
<td>Startide Rising</td>
</tr>
<tr>
<td>James Gleick</td>
<td>Chaos: Making a New Science</td>
</tr>
<tr>
<td>Charles Dickens</td>
<td>Great Expectations</td>
</tr>
<tr>
<td>William Shakespeare</td>
<td>The Comedy of Errors</td>
</tr>
</tbody>
</table>

- Learn these patterns:

<table>
<thead>
<tr>
<th>URL Prefix</th>
<th>Text Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.sff.net/locus/c">www.sff.net/locus/c</a>.*</td>
<td>`&lt;LI&gt;&lt;B&gt;title&lt;/B&gt; by author (</td>
</tr>
<tr>
<td>dns.city-net.com/~lmann/awards/hugs/1984.html</td>
<td>`&lt;i&gt;title&lt;/i&gt; by author (</td>
</tr>
<tr>
<td>dolphin.upenn.edu/~dcummins/texts/sf-award.htm</td>
<td>author</td>
</tr>
</tbody>
</table>

- Now iterate, using these patterns to get more instances and patterns...
Snowball (Agichtein & Gravano 2000)

New idea: require that X and Y be named entities of particular types

<table>
<thead>
<tr>
<th>Organization</th>
<th>Location of Headquarters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft</td>
<td>Redmond</td>
</tr>
<tr>
<td>Exxon</td>
<td>Irving</td>
</tr>
<tr>
<td>IBM</td>
<td>Armonk</td>
</tr>
<tr>
<td>Boeing</td>
<td>Seattle</td>
</tr>
<tr>
<td>Intel</td>
<td>Santa Clara</td>
</tr>
</tbody>
</table>

New idea: require that X and Y be named entities of particular types.
Bootstrapping problems

- Requires that we have seeds for each relation
  - Sensitive to original set of seeds
- Big problem of semantic drift at each iteration
- Precision tends to be not that high
- Generally have lots of parameters to be tuned
- Don’t have a probabilistic interpretation
  - Hard to know how confident to be in each result
5 easy methods for relation extraction

1. Hand-built patterns
2. Supervised methods
3. Bootstrapping (seed) methods
4. **Unsupervised methods**
5. Distant supervision
KnowItAll (Etzioni et al. 2005)

- Input: target class labels and relation labels

<table>
<thead>
<tr>
<th>Predicate</th>
<th>class label</th>
<th>relation label</th>
</tr>
</thead>
<tbody>
<tr>
<td>City</td>
<td>“city”, “town”</td>
<td>___</td>
</tr>
<tr>
<td>Country</td>
<td>“country”, “nation”</td>
<td>___</td>
</tr>
<tr>
<td>capitalOf(City,Country)</td>
<td>___</td>
<td>“capital of”</td>
</tr>
</tbody>
</table>

- Use Hearst patterns to find instances of classes
  - ENTITY and/or other CLASS, such CLASS as ENTITY, etc

- Now use new pattern templates to find relations
  - CLASS1 is the RELATION CLASS2
  - CLASS1, RELATION CLASS2

- So once you learn Paris and Berlin are cities
  - Can use “Paris is the capital of France” to extract capitalOf(Paris, France)
KnowItAll PMI-based Assessor

- Validate candidate instances using “discriminators”
- Compare # search engine hits for
  - instance alone
  - Instance + discriminator

\[
PMI(I, D) = \frac{|\text{Hits}(D + I)|}{|\text{Hits}(I)|}
\]

- (This is \emph{not} the conventional definition of PMI!)
- Use “PMI” scores as features for Naïve Bayes classifier

\textbf{Example: linguists such as}

- \( \text{PMI(linguists such as, Chomsky)} = \frac{4000}{17.5\text{M}} = 2.23 \text{E-04} \)
- \( \text{PMI(linguists such as, Potts)} = \frac{1}{26.8\text{M}} = 3.73 \text{E-08} \)
TextRunner (Banko et al. 2007)

1. **Self-Supervised Learner**: automatically labels +/- examples & learns an extractor

2. **Single-Pass Extractor**: single pass over corpus, identifying extractions in each sentence

3. **Redundancy-Based Assessor**: Assign a probability to each extraction
Step 1: Self-Supervised Learner

• Run a parser over 2000 sentences
  • expensive (0.5 seconds/parse) so can’t run on whole web
  • For each pair of base noun phrases NP_i and NP_j
  • Extract all tuples t = (NP_i, relation_{i,j}, NP_j)

• Now label each tuple t as positive if and only if:
  • The dependency path between entities is short
  • The dependency path doesn’t cross a clause boundary
  • Neither NP is a pronoun

• Now train a Naïve Bayes classifier to distinguish them
  • using features like POS tags nearby, stop words, etc. etc.
Step 2: Single-Pass Extractor

Over a huge (web-sized) corpus:

- Run a dumb POS tagger
- Run a dumb Base Noun Phrase finder
- Extract all text strings between base NPs
- Run heuristic rules to simplify text strings
  
  *Scientists from many universities are intently studying stars*
  
  $\rightarrow \langle\text{scientists, are studying, stars}\rangle$

- Pass candidate tuple to classifier
- Save only those predicted to be “trustworthy”
Step 3: Redundancy-Based Assessor

- Collect counts for each simplified relation
  \[ \langle \text{scientists}, \text{are studying}, \text{stars} \rangle \rightarrow 17 \]

- Given the counts for each relation, and the number of sentences, they use a combinatoric balls-and-urns model to compute probability of each relation [Downey et al. 05]

\[
P(x \in C | x \text{ appears } k \text{ times in } n \text{ draws}) \approx \frac{1}{1 + \frac{|E|}{|C|} \left( \frac{p_E}{p_C} \right)^k e^n (p_C - p_E)}
\]
TextRunner demo

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

(Note that they’ve re-branded TextRunner as ReVerb, but it’s essentially the same as before.)
## TextRunner examples

<table>
<thead>
<tr>
<th>Probability</th>
<th>Count</th>
<th>Arg1</th>
<th>Predicate</th>
<th>Arg2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.98</td>
<td>59</td>
<td>Smith</td>
<td>invented</td>
<td>the margherita</td>
</tr>
<tr>
<td>0.97</td>
<td>49</td>
<td>Al Gore</td>
<td>invented</td>
<td>the Internet</td>
</tr>
<tr>
<td>0.97</td>
<td>44</td>
<td>manufacturing plant</td>
<td>first invented</td>
<td>the automatic revolver</td>
</tr>
<tr>
<td>0.97</td>
<td>41</td>
<td>Alexander Graham Bell</td>
<td>invented</td>
<td>the telephone</td>
</tr>
<tr>
<td>0.97</td>
<td>36</td>
<td>Thomas Edison</td>
<td>invented</td>
<td>light bulbs</td>
</tr>
<tr>
<td>0.97</td>
<td>29</td>
<td>Eli Whitney</td>
<td>invented</td>
<td>the cotton gin</td>
</tr>
<tr>
<td>0.96</td>
<td>23</td>
<td>C. Smith</td>
<td>invented</td>
<td>the margherita</td>
</tr>
<tr>
<td>0.96</td>
<td>19</td>
<td>the Digital Equipment Corporation manufacturing plant</td>
<td>first invented</td>
<td>the automatic revolver</td>
</tr>
<tr>
<td>0.96</td>
<td>18</td>
<td>Edison</td>
<td>invented</td>
<td>the phonograph</td>
</tr>
</tbody>
</table>
Results from TextRunner

- From corpus of 9M web pages, containing 133M sentences
- Extracted 60.5 million tuples
  - \( \langle FCI, \text{specializes in, software development} \rangle \)
- Evaluation
  - Not well formed:
    - \( \langle \text{demands, of securing, border} \rangle, \langle 29, \text{dropped, instruments} \rangle \)
  - Abstract:
    - \( \langle \text{Einstein, derived, theory} \rangle, \langle \text{executive, hired by, company} \rangle \)
  - True, concrete:
    - \( \langle \text{Tesla, invented, coil transformer} \rangle \)
Evaluating TextRunner

figure from Banko et al. 2007
5 easy methods for relation extraction

1. Hand-built patterns
2. Supervised methods
3. Bootstrapping (seed) methods
4. Unsupervised methods
5. Distant supervision
Distant supervision paradigm

Snow, Jurafsky, Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. NIPS 17


• Hypothesis: If two entities are known to belong to a certain relation, any sentence containing those two entities is likely to express that relation.

• Distant supervision: use *database* of relations to get lots of training examples
  • instead of hand-creating a few seed tuples (bootstrapping)
  • instead of using hand-labeled corpus (supervised)
Distant supervision paradigm

- For each pair of entities in a large database:
  - Grab sentences containing these entities from a corpus
  - Extract lots of noisy features from the sentences
    - Lexical features, syntactic features, named entity tags
  - Combine in a classifier
Distant supervision paradigm

- Has advantages of supervised approach:
  - use of rich of hand-created knowledge
  - relations have canonical names
  - can use rich features (e.g. syntactic features)

- Has advantages of unsupervised approach:
  - unlimited amounts of data
  - allows for very large number of weak features
  - not sensitive to training corpus: genre-independent
We construct a noisy training set consisting of occurrences from our corpus that contain an IS-A pair according to WordNet.

This yields high-signal examples like:
“...consider authors like Shakespeare...”
“Some authors (including Shakespeare)...”
“Shakespeare was the author of several...”
“Shakespeare, author of The Tempest...”
We construct a noisy training set consisting of occurrences from our corpus that contain an IS-A pair according to WordNet. This yields high-signal examples like:

“...consider authors like Shakespeare...”
“Some authors (including Shakespeare)...”
“Shakespeare was the author of several...”
“Shakespeare, author of The Tempest...”

But also noisy examples like:

“The author of Shakespeare in Love...”
“...authors at the Shakespeare Festival...”

Training set (TREC and Wikipedia):
14,000 hypernym pairs, ~600,000 total pairs
Learning patterns


1. Take corpus sentences ...
2. Collect noun pairs
   752,311 pairs from 6M words of newswire
3. Is pair an IS-A in WordNet?
   14,387 yes, 737,924 no
4. Parse the sentences
5. Extract patterns
   69,592 dependency paths >5 pairs)
6. Train classifier on these patterns
   Logistic regression with 70K features (actually converted to 974,288 bucketed binary features)
One of 70,000 patterns

“<superordinate> ‘called’ <subordinate>”

Learned from cases such as:

“sarcoma / cancer”: ...an uncommon bone cancer called osteogenic sarcoma and to...
“deuterium / atom” ....heavy water rich in the doubly heavy hydrogen atom called deuterium.

New pairs discovered:

“efflorescence / condition”: ...and a condition called efflorescence are other reasons for...
“’neal_inc / company” ...The company, now called O'Neal Inc., was sole distributor of E-Ferol...
“hat_creek_outfit / ranch” ...run a small ranch called the Hat Creek Outfit.
“hiv-1 / aids_virus” ...infected by the AIDS virus, called HIV-1.
“bateau_mouche / attraction” ...local sightseeing attraction called the Bateau Mouche...
“kibbutz_malkiyya / collective_farm” ...an Israeli collective farm called Kibbutz Malkiyya...
Recording the Lexico-Syntactic Environment with Syntactic Dependency Paths

MINIPAR: A principle-based dependency parser (Lin, 1998)

Example Word Pair: “Shakespeare / author”
Example Sentence: “Shakespeare was the author of several plays…”

Minipar Parse:
MINIPAR: A principle-based dependency parser (Lin, 1998)

Example Word Pair: “Shakespeare / author”

Example Sentence: “Shakespeare was the author of several plays...”

Minipar Parse:

- be
  - Shakespeare
  - author

Extract shortest path:

“-N:s:VBE, “be”, VBE:pred:N”
Hearst patterns to MINIPAR dependency paths

**Hearst Pattern**

- Y such as X...
- Such Y as X...
- X... and other Y

**MINIPAR Representation**

1. Y such as X...
   - N:mod:Prep, such_as, Prep:pcomp-n:N, X
   - N:pcomp-n:Prep, such_as, such_as, Prep:mod:N

2. Such Y as X...
   - Y as Prep:pcomp-n:N, X
   - N:mod:Prep, as, as, Prep:mod:N, {such, PreDet:pre:N}

3. X... and other Y
   - X Y other
Hypernym Precision / Recall for all Features

Individual Feature Analysis

Y including X

Slide from Rion Snow
Precision/recall for various patterns

Individual Feature Analysis

- Y including X
- Y such as X
Precision/recall for various patterns

Individual Feature Analysis

- □ Y including X
- ✦ Y such as X
- ✗ X and/or other Y

Precision

Recall (log)
Precision/recall for various patterns

Individual Feature Analysis

- X and/or other Y
- Y such as X
- such Y as X
- Y including X
- Y, especially X
- Y like X
- Y called X
- X is Y
- X, a Y (appositive)
Precision/recall for hypernym classifier

logistic regression

$$P(R|E) = \frac{1}{1 + e^{-\sum w_i x_i}}$$

10-fold Cross Validation on 14,000 WordNet-Labeled Pairs

Hypernym Classifiers on WordNet-labeled dev set

- Best Logistic Regression (Buckets): 0.3480
- Best Logistic Regression (Binary): 0.3200
- Best Multinomial Naive Bayes: 0.3175
- Best Complement Naive Bayes: 0.3024
- Hearst Patterns: 0.1500
- "And/Or Other" Pattern: 0.1170

F-score

slide from Rion Snow
What about other relations?


Training Set

- Freebase
- 102 relations
- 940,000 entities
- 1.8 million instances

Corpus

- Wikipedia
- 1.8 million articles
- 25.7 million sentences

Slide adapted from Rion Snow
# Frequent Freebase Relations

<table>
<thead>
<tr>
<th>Relation name</th>
<th>Size</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>/people/person/nationality</td>
<td>281,107</td>
<td>John Dugard, South Africa</td>
</tr>
<tr>
<td>/location/location/contains</td>
<td>253,223</td>
<td>Belgium, Nijlen</td>
</tr>
<tr>
<td>/people/person/profession</td>
<td>208,888</td>
<td>Dusa McDuff, Mathematician</td>
</tr>
<tr>
<td>/people/person/place_of_birth</td>
<td>105,799</td>
<td>Edwin Hubble, Marshfield</td>
</tr>
<tr>
<td>/dining/restaurant/cuisine</td>
<td>86,213</td>
<td>MacAyo’s Mexican Kitchen, Mexican</td>
</tr>
<tr>
<td>/business/business_chain/location</td>
<td>66,529</td>
<td>Apple Inc., Apple Inc., South Park, NC</td>
</tr>
<tr>
<td>/biology/organism_classification_rank</td>
<td>42,806</td>
<td>Scorpaeniformes, Order</td>
</tr>
<tr>
<td>/film/film/genre</td>
<td>40,658</td>
<td>Where the Sidewalk Ends, Film noir</td>
</tr>
<tr>
<td>/film/film/language</td>
<td>31,103</td>
<td>Enter the Phoenix, Cantonese</td>
</tr>
<tr>
<td>/biology/organism_higher_classification</td>
<td>30,052</td>
<td>Calopteryx, Calopterygida</td>
</tr>
<tr>
<td>/film/film/country</td>
<td>27,217</td>
<td>Turtle Diary, United States</td>
</tr>
<tr>
<td>/film/writer/film</td>
<td>23,856</td>
<td>Irving Shulman, Rebel Without a Cause</td>
</tr>
<tr>
<td>/film/director/film</td>
<td>23,539</td>
<td>Michael Mann, Collateral</td>
</tr>
<tr>
<td>/film/producer/film</td>
<td>22,079</td>
<td>Diane Eskenazi, Aladdin</td>
</tr>
<tr>
<td>/people/deceased_person/place_of_death</td>
<td>18,814</td>
<td>John W. Kern, Asheville</td>
</tr>
<tr>
<td>/music/artist/origin</td>
<td>18,619</td>
<td>The Octopus Project, Austin</td>
</tr>
<tr>
<td>/people/person/religion</td>
<td>17,582</td>
<td>Joseph Chartrand, Catholicism</td>
</tr>
<tr>
<td>/book/author/works-written</td>
<td>17,278</td>
<td>Paul Auster, Travels in the Scriptorium</td>
</tr>
<tr>
<td>/soccer/football_position/players</td>
<td>17,244</td>
<td>Midfielder, Chen Tao</td>
</tr>
<tr>
<td>/people/deceased_person/cause_of_death</td>
<td>16,709</td>
<td>Richard Daintree, Tuberculosis</td>
</tr>
<tr>
<td>/film/film/music</td>
<td>14,070</td>
<td>Stavisky, Stephen Sondheim</td>
</tr>
<tr>
<td>/business/company/industry</td>
<td>13,805</td>
<td>ATS Medical, Health care</td>
</tr>
</tbody>
</table>
Bill Gates founded Microsoft in 1975.
Bill Gates, founder of Microsoft, …
Bill Gates attended Harvard from…
Google was founded by Larry Page and…

**Text**

**Freebase relations**

Founder: <Bill Gates, Microsoft>
Founder: <Larry Page, Google>
CollegeAttended: <Bill Gates, Harvard>
Bill Gates founded Microsoft in 1975.
Bill Gates, founder of Microsoft, …
Bill Gates attended Harvard from…
Google was founded by Larry Page and…

**Freebase relations**

- Founder: <Bill Gates, Microsoft>
- Founder: <Larry Page, Google>
- CollegeAttended: <Bill Gates, Harvard>

**Extracted training data**

- [Founder]
  - “X founded Y”
Bill Gates founded Microsoft in 1975. Bill Gates, founder of Microsoft, … Bill Gates attended Harvard from… Google was founded by Larry Page and…

Extracted training data

[Founder]

• “X founded Y”
• “X, founder of Y”

Freebase relations

Founder: <Bill Gates, Microsoft>
Founder: <Larry Page, Google>
CollegeAttended: <Bill Gates, Harvard>
Bill Gates founded Microsoft in 1975.
Bill Gates, founder of Microsoft, …
**Bill Gates attended Harvard from**…
Google was founded by Larry Page and…

**Freebase relations**

- Founder: `<Bill Gates, Microsoft>`
- Founder: `<Larry Page, Google>`
- CollegeAttended: `<Bill Gates, Harvard>`

**Extracted training data**

- [Founder]
  - "X founded Y"
  - "X, founder of Y"
- [CollegeAttended]
  - "X attended Y"
Bill Gates founded Microsoft in 1975.
Bill Gates, founder of Microsoft, …
Bill Gates attended Harvard from…

Google was founded by Larry Page …

Founder: <Bill Gates, Microsoft>
Founder: <Larry Page, Google>
CollegeAttended: <Bill Gates, Harvard>

Extracted training data

[Founder]
• “X founded Y”
• “X, founder of Y”
• “Y was founded by X”

[CollegeAttended]
• “X attended Y”
Training

Extracted training data

[Founder]
• “X founded Y”
• “X, founder of Y”
• “Y was founded by X”

[CollegeAttended]
• “X attended Y”

Train \[ \rightarrow \] Relation Classifier
Testing

Corpus text

Henry Ford founded Ford Motor Co. in...
Ford Motor Co. was founded by Henry Ford...
Steve Jobs attended Reed College from...

Extracted testing data
Henry Ford founded Ford Motor Co. in...
Ford Motor Co. was founded by Henry Ford...
Steve Jobs attended Reed College from...

<Henry Ford, Ford Motor Co.>
[??]
- “X founded Y”
Testing

Corpus text

Henry Ford founded Ford Motor Co. in…
**Ford Motor Co. was founded by Henry Ford**…
Steve Jobs attended Reed College from…

Extracted testing data

<Henry Ford, Ford Motor Co.>
[???
• “X founded Y”
• “Y was founded by X”
Testing

Corpus text

Henry Ford founded Ford Motor Co. in…
Ford Motor Co. was founded by Henry Ford…
**Steve Jobs attended Reed College from…**

Extracted testing data

<table>
<thead>
<tr>
<th>&lt;Henry Ford, Ford Motor Co.&gt;</th>
<th>&lt;Steve Jobs, Reed College&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>[???]</td>
<td>[???]</td>
</tr>
<tr>
<td>•“X founded Y”</td>
<td>•“X attended Y”</td>
</tr>
<tr>
<td>•“Y was founded by X”</td>
<td></td>
</tr>
</tbody>
</table>
Extracted testing data

• “X founded Y”
• “Y was founded by X”

• “X attended Y”

Results!

<Henry Ford, Ford Motor Co.>: Founder
<Steve Jobs, Reed College>: CollegeAtt.
Advantage

• ACE paradigm: labeling sentences

• Our paradigm: labeling entity pairs
  • We make use of multiple appearances of entities
  • If a pair of entities appears in 10 sentences, and each sentence has 5 features extracted from it, the entity pair will have 50 associated features
Lexical and Syntactic Features

Astronomer Edwin Hubble was born in Marshfield, Missouri

<table>
<thead>
<tr>
<th>Feature type</th>
<th>Left window</th>
<th>NE1</th>
<th>Middle</th>
<th>NE2</th>
<th>Right window</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexical</td>
<td>[]</td>
<td>PER</td>
<td>was/VERB born/VERB in/CLOSED</td>
<td>LOC</td>
<td>[]</td>
</tr>
<tr>
<td>Lexical</td>
<td>[Astronomer]</td>
<td>PER</td>
<td>was/VERB born/VERB in/CLOSED</td>
<td>LOC</td>
<td>[. Missouri]</td>
</tr>
<tr>
<td>Lexical</td>
<td>[#PAD#, Astronomer]</td>
<td>PER</td>
<td>was/VERB born/VERB in/CLOSED</td>
<td>LOC</td>
<td>[]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[]</td>
<td>PER</td>
<td>(\uparrow) was (\downarrow) pred born (\downarrow) mod in (\downarrow) pcomp-n</td>
<td>LOC</td>
<td>[]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[Edwin Hubble (\downarrow) lex-mod]</td>
<td>PER</td>
<td>(\uparrow) was (\downarrow) pred born (\downarrow) mod in (\downarrow) pcomp-n</td>
<td>LOC</td>
<td>[]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[Astronomer (\downarrow) lex-mod]</td>
<td>PER</td>
<td>(\uparrow) was (\downarrow) pred born (\downarrow) mod in (\downarrow) pcomp-n</td>
<td>LOC</td>
<td>[]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[]</td>
<td>PER</td>
<td>(\uparrow) was (\downarrow) pred born (\downarrow) mod in (\downarrow) pcomp-n</td>
<td>LOC</td>
<td>[(\downarrow) lex-mod , ]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[Edwin Hubble (\downarrow) lex-mod]</td>
<td>PER</td>
<td>(\uparrow) was (\downarrow) pred born (\downarrow) mod in (\downarrow) pcomp-n</td>
<td>LOC</td>
<td>[(\downarrow) lex-mod , ]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[Astronomer (\downarrow) lex-mod]</td>
<td>PER</td>
<td>(\uparrow) was (\downarrow) pred born (\downarrow) mod in (\downarrow) pcomp-n</td>
<td>LOC</td>
<td>[(\downarrow) lex-mod , ]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[]</td>
<td>PER</td>
<td>(\uparrow) was (\downarrow) pred born (\downarrow) mod in (\downarrow) pcomp-n</td>
<td>LOC</td>
<td>[(\downarrow) inside Missouri]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[Edwin Hubble (\downarrow) lex-mod]</td>
<td>PER</td>
<td>(\uparrow) was (\downarrow) pred born (\downarrow) mod in (\downarrow) pcomp-n</td>
<td>LOC</td>
<td>[(\downarrow) inside Missouri]</td>
</tr>
<tr>
<td>Syntactic</td>
<td>[Astronomer (\downarrow) lex-mod]</td>
<td>PER</td>
<td>(\uparrow) was (\downarrow) pred born (\downarrow) mod in (\downarrow) pcomp-n</td>
<td>LOC</td>
<td>[(\downarrow) inside Missouri]</td>
</tr>
</tbody>
</table>

Diagram:

```
Astronomer   Edwin Hubble   was   born   in Marshfield, Missouri
```

- **lex-mod**: Logical relation indicating the profession of Edwin Hubble.
- **s**: The subject of the sentence, Edwin Hubble.
- **pred**: The predicate, "was born in Marshfield, Missouri".
- **mod**: Modifier, "in".
- **pcomp-n**: Prepositional phrase ending with "in".
- **inside**: Indicates a location.

The diagram shows the syntactic structure of the sentence, highlighting the relationships between the noun, verb, and location components.
Examples of high-weight features

<table>
<thead>
<tr>
<th>Relation</th>
<th>Feature type</th>
<th>Left window</th>
<th>NE1</th>
<th>Middle</th>
<th>NE2</th>
<th>Right window</th>
</tr>
</thead>
<tbody>
<tr>
<td>/architecture/structure/architect</td>
<td>LEX</td>
<td></td>
<td>ORG</td>
<td>the designer of the</td>
<td>PER</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYN</td>
<td>designed</td>
<td>s</td>
<td>by</td>
<td>PER</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>story</td>
<td>is</td>
<td>ORG</td>
<td></td>
</tr>
<tr>
<td>/book/author/works_written</td>
<td>LEX</td>
<td></td>
<td>PER</td>
<td>s novel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/business/company/founders</td>
<td>LEX</td>
<td></td>
<td>ORG</td>
<td>co-founder</td>
<td>PER</td>
<td></td>
</tr>
<tr>
<td>/business/company/place_founded</td>
<td>LEX</td>
<td></td>
<td>ORG</td>
<td>based</td>
<td>LOC</td>
<td></td>
</tr>
<tr>
<td>/film/film/country</td>
<td>LEX</td>
<td></td>
<td>PER</td>
<td>released in</td>
<td>LOC</td>
<td></td>
</tr>
<tr>
<td>/geography/river/mouth</td>
<td>LEX</td>
<td></td>
<td>LOC</td>
<td>which flows into the</td>
<td>LOC</td>
<td></td>
</tr>
<tr>
<td>/government/political_party/country</td>
<td>LEX</td>
<td></td>
<td>ORG</td>
<td>politician of the</td>
<td>LOC</td>
<td></td>
</tr>
<tr>
<td>/influence/influence_node/influenced</td>
<td>LEX</td>
<td></td>
<td>PER</td>
<td>a student of</td>
<td>LOC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYN</td>
<td>candidate</td>
<td>ORG</td>
<td>tributary</td>
<td>LOC</td>
<td></td>
</tr>
<tr>
<td>/language/human_language/region</td>
<td>LEX</td>
<td></td>
<td>PER</td>
<td>student</td>
<td>PER</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYN</td>
<td>of</td>
<td>ORG</td>
<td>speaking areas of</td>
<td>LOC</td>
<td></td>
</tr>
<tr>
<td>/music/artist/origin</td>
<td>LEX</td>
<td></td>
<td>ORG</td>
<td>based band</td>
<td>LOC</td>
<td></td>
</tr>
<tr>
<td>/people/deceased_person/place_of_death</td>
<td>LEX</td>
<td></td>
<td>PER</td>
<td>died in</td>
<td>LOC</td>
<td></td>
</tr>
<tr>
<td>/people/person/nationality</td>
<td>LEX</td>
<td></td>
<td>PER</td>
<td>is a citizen of</td>
<td>LOC</td>
<td></td>
</tr>
<tr>
<td>/people/person/parents</td>
<td>LEX</td>
<td></td>
<td>PER</td>
<td>from</td>
<td>LOC</td>
<td></td>
</tr>
<tr>
<td>/people/person/place_of_birth</td>
<td>LEX</td>
<td></td>
<td>PER</td>
<td>is the birthplace of</td>
<td>PER</td>
<td></td>
</tr>
<tr>
<td>/people/person/religion</td>
<td>LEX</td>
<td></td>
<td>PER</td>
<td>embraced</td>
<td>LOC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYN</td>
<td>convert</td>
<td>PER</td>
<td>convert</td>
<td>LOC</td>
<td></td>
</tr>
</tbody>
</table>
Implementation

• Classifier: multi-class logistic regression optimized using L-BFGS with Gaussian regularization (Manning & Klein 2003)

• Parser: MINIPAR (Lin 1998)

• POS tagger: MaxEnt tagger trained on the Penn Treebank (Toutanova et al. 2003)

• NER tagger: Stanford four-class tagger \{person, location, organization, miscellaneous, none\} (Finkel et al. 2005)

• 3 configurations: lexical features, syntax features, both
Experiment

- 1.8 million relation instances used for training
  - Compared to 17,000 relation instances in ACE

- 800,000 Wikipedia articles used for training, 400,000 different articles used for testing

- We only extract relation instances that are not already in Freebase
Newly discovered relation instances

<table>
<thead>
<tr>
<th>Relation name</th>
<th>New instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>/location/location/contains</td>
<td>Paris, Montmartre</td>
</tr>
<tr>
<td>/location/location/contains</td>
<td>Ontario, Fort Erie</td>
</tr>
<tr>
<td>/music/artist/origin</td>
<td>Mighty Wagon, Cincinnati</td>
</tr>
<tr>
<td>/people/deceased_person/place_of_death</td>
<td>Fyodor Kamensky, Clearwater</td>
</tr>
<tr>
<td>/people/person/nationality</td>
<td>Marianne Yvonne Heemskerk, Netherlands</td>
</tr>
<tr>
<td>/people/person/place_of_birth</td>
<td>Wavell Wayne Hinds, Kingston</td>
</tr>
<tr>
<td>/book/author/works_written</td>
<td>Upton Sinclair, Lanny Budd</td>
</tr>
<tr>
<td>/business/company/founders</td>
<td>WWE, Vince McMahon</td>
</tr>
<tr>
<td>/people/person/profession</td>
<td>Thomas Mellon, judge</td>
</tr>
</tbody>
</table>

Ten relation instances extracted by the system that did not appear in Freebase
Evaluation

• Held-out evaluation
  • Train on 50% of gold-standard Freebase relation instances, test on other 50%
  • Used to tune parameters quickly without having to wait for human evaluation

• Human evaluation
  • Performed by evaluators on Amazon Mechanical Turk
  • Calculated precision at 100 and 1000 recall levels for the 10 most common relations
Held-out evaluation

Automatic evaluation on 900K instances of 102 Freebase relations. Precision for three different feature sets is reported at various recall levels.
Human evaluation

Precision, using Mechanical Turk labelers:

<table>
<thead>
<tr>
<th>Relation name</th>
<th>100 instances</th>
<th>1000 instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>/film/director/film</td>
<td>0.49</td>
<td>0.49</td>
</tr>
<tr>
<td>/film/writer/film</td>
<td>0.70</td>
<td>0.71</td>
</tr>
<tr>
<td>/geography/river/basin_countries</td>
<td>0.65</td>
<td>0.73</td>
</tr>
<tr>
<td>/location/country/administrative_divisions</td>
<td>0.68</td>
<td>0.72</td>
</tr>
<tr>
<td>/location/location/contains</td>
<td>0.81</td>
<td>0.85</td>
</tr>
<tr>
<td>/location/us_county/county-seat</td>
<td>0.51</td>
<td>0.47</td>
</tr>
<tr>
<td>/music/artist/origin</td>
<td>0.64</td>
<td>0.61</td>
</tr>
<tr>
<td>/people/deceased_person/place_of_death</td>
<td>0.80</td>
<td>0.80</td>
</tr>
<tr>
<td>/people/person/nationality</td>
<td>0.61</td>
<td>0.56</td>
</tr>
<tr>
<td>/people/person/place_of_birth</td>
<td>0.78</td>
<td>0.88</td>
</tr>
<tr>
<td>Average</td>
<td>0.67</td>
<td>0.68</td>
</tr>
</tbody>
</table>

Syn | Lex | Both | Syn | Lex | Both |
--- | --- | ---  | --- | --- | ---  |
0.49 | 0.43 | 0.44 | 0.49 | 0.41 | 0.46 |
0.70 | 0.60 | 0.65 | 0.71 | 0.61 | 0.69 |
0.65 | 0.64 | 0.67 | 0.73 | 0.71 | 0.64 |
0.68 | 0.59 | 0.70 | 0.72 | 0.68 | 0.72 |
0.81 | 0.89 | 0.84 | 0.85 | 0.83 | 0.84 |
0.51 | 0.51 | 0.53 | 0.47 | 0.57 | 0.42 |
0.64 | 0.66 | 0.71 | 0.61 | 0.63 | 0.60 |
0.80 | 0.79 | 0.81 | 0.80 | 0.81 | 0.78 |
0.61 | 0.70 | 0.72 | 0.56 | 0.61 | 0.63 |
0.78 | 0.77 | 0.78 | 0.88 | 0.85 | 0.91 |
0.67 | 0.66 | 0.69 | 0.68 | 0.67 | 0.67 |
Human evaluation

• At recall of 100 instances, using both feature sets (lexical and syntax) offers the best performance for a majority of the relations

• At recall of 1000 instances, using syntax features improves performance for a majority of the relations
Where syntax helps

*Back Street* is a 1932 film made by Universal Pictures, directed by **John M. Stahl**, and produced by Carl Laemmle Jr.

*Back Street* and **John M. Stahl** are far apart in surface string but close together in dependency parse.
Where syntax doesn’t help

Beaverton is a city in Washington County, Oregon ...

Beaverton and Washington County are close together in surface string
Conclusions

• Distant supervision extracts high-precision patterns for a variety of relations.

• Can make use of 1000 times more data than simple supervised algorithms.

• Syntax features almost always help.

• The combination of syntax and lexical features is sometimes even better.

• Syntax features are probably most useful when entities are far apart, often when there are modifiers in between.
Discussion

- Relation extraction $\rightarrow$ learning by reading
- Suppose we could do relation extraction perfectly?
- What would we still be missing?
- What knowledge could we still not gather from the web?