Reminders

• Next week is spring break, no class
  – Office hour will be by appointment only

• Midterm report due on 5p ET March 21st
  – An hour before the class
  – See last week’s lecture note for late submission policy

• Any other logistic question?
Outline

• Information extraction systems
  – (Snowball)
  – Cimple
  – KnowItAll/TextRunner

• Quiz + Break (30 min)

• Recent research
  – Prioritization of information extraction
  – Enhancing Wikipedia

• Conclusion
Recap of Last Lecture

• Main tasks of information extraction

• Basic techniques
  – Named entity recognition
  – Wrapper induction

• System:
  – Snowball
Review of Snowball

Seed Tuples → Find Occurrences of Seed Tuples

Generate New Seed Tuples

Snowball

Tag Entities

Augment Table ← Generate Extraction Patterns
Landscape of Extraction Systems

Tools and Platforms:
- IBM UIMA
- Open Calais

Low Manual Effort:
- Open Web
  - TextRunner
- Domain-centric
  - Snowball

Modest Manual Effort:
- Cimple
Landscape of Extraction Systems

Open Web

Low Manual Effort

TextRunner

Domain-centric

Snowball

Modest Manual Effort

Cimple
Goal of Cimple

- Community Information Management: building comprehensive structured community portals with modest amount of human efforts

- Community: a group of users who share similar interests and a common ontology
  - I.e., domain centric

- Characteristics
  - Top-down
  - Compositional
  - Incremental
Top-down

• Start by focusing on a few well-known Web sources within the community
  – Database research community: DBWorld, DBLP, etc.
  – Entertainment community: IMDb, Rotten Tomatoes, etc.

• This initial set is small enough such that building an extraction plan *manually* is relatively easy
Compositional

- Dividing the complex extraction task into individual small tasks
- Each small task is simple enough to be carried out by easy-to-implement operators
- The results are consolidated in the end
- Enables declarative information extraction
Incremental

- Simple assumption: important data sources will sooner or later show up in the set of sources currently being monitored

- Therefore, there is no need to actively go out and crawl the Web for additional sources
Top-down: Initial Source Ranking

- **80/20 Rules**
  - 20% of the sources covers 80% of the community knowledge
  - How to identify the 20%?
    - Collect as many as possible and rank them!

- **Ranking methods**
  - PageRank
  - PageRank + virtual link
  - PageRank + virtual link + tf*idf: winner!

- **Virtual link:**
  - Add a link between two sources if they contain pages that mention the same entity
TF*IDF

- A frequently used technique in information retrieval
  - TF: term frequency
    - Representing how often a term appears in a document
  - IDF: inverted document frequency
    - Representing how rarely a term appears in over the entire corpus

- Given a term t, a document d, over a corpus C:
  - Higher tf(t,d) means t is more relevant to d
  - Higher idf(t,C) means t is more specific to the documents it appears in (e.g., d)

- In this work:
  \[
  \text{TF}(e, s) = \frac{c(e)}{\sum_{f \in E_s} c(f)},
  \]
  \[
  \text{IDF}(e, S) = \log \frac{|S|}{|\{t \in S : e \in E_t\}|}
  \]
Compositional: Building the Initial Portal

• Starts with a predefined schema that defines
  – Entity types
  – Relation types
• Define and implement a set of operators
• Write extraction plans to connect operators
• Think of database operators!
Entity Operators

- **ExtractM**
  - Detect mentions of a given type
  - Named entity recognizer!
  - ExtractMbyName
    - A dictionary-based entity recognizer
- **MatchM**
  - Match mentions that refer to the same real world entity
  - MatchMbyName, MatchMStrict
    - Matching two mentions purely based on name similarity (accounting for common name variants)
- **CreateE**
  - Create an entity for each group of matched mentions

- **Both ExtractM and MatchM are difficult research topics, Cimple’s claim is that simple methods is enough for Web communities**
Relation Operators

• **ComputeCoStrength**
  – Given a pair of entities, compute the strength of their co-occurrences in the sources
    • Two entities with high co-occurrence scores are considered to be related
    • The co-occurrences can be constrained in the context
  – The type of the resulting relations depends on the types of the input entities
    • $R_{\text{served}}(E_{\text{person}}, E_{\text{conference}})$, $R_{\text{published}}(E_{\text{person}}, E_{\text{paper}})$

• **CreateR**
  – Create a relation for each pair of related entities

• **ExtractLabel** *(utility operator)*
  – Given a set of pages, produce a set of labels
Extraction Plans

CreateE
\[\text{MatchMStrict}\]
\[R\]
\[\text{MatchMbyName}\]
\[\text{ExtractMbyName}\]
\[\text{ExtractMbyName}\]
\[\text{Union}\]
\[\{s_1 \ldots s_n\} \setminus \text{DBLP}\]
\[\text{DBLP}\]
Extraction Plans

CreateR
  \mid Select(strength > \theta)
  \mid ComputeCoStrength
  \mid \times
  \mid \text{Union}
  \bullet person entities
  \bullet org entities
  \bullet s_1 \ldots s_n

Affiliation

CreateR
  \mid c(person, label)
  \mid \text{ExtractLabel}
  \mid \text{main pages}
  \bullet person entities
  \bullet conference entities

PC membership
Integration

- Each plan generates a fragment of the final ER graph
  - Operators MatchE/EnrichE then consolidate those fragments
- Integration is needed in two aspects
  - Integrate ER fragments from multiple sub-plans in the same daily plan
  - Integrate results from a daily plan into the existing ER graph generated earlier
## Some Statistics

<table>
<thead>
<tr>
<th></th>
<th>Initial</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sources</td>
<td>846</td>
<td>1,075</td>
</tr>
<tr>
<td>Entities</td>
<td>489 [5 types]</td>
<td>16,674</td>
</tr>
<tr>
<td>Relations</td>
<td>[8 types]</td>
<td>63,923</td>
</tr>
</tbody>
</table>
Summary

• Declarative information extraction supported by a suite of extraction operators is a promising direction
  – Easy to produce extraction tasks
  – Easy to debug and evaluate

• The Cimple project is a step toward that direction, but much remains to be done
KnowItAll / TextRunner

Landscape of Extraction Systems

Low Manual Effort

Open Web
- TextRunner

Domain-centric
- Snowball

Modest Manual Effort

- Cimple
Main Goal

- Leverage the entire Web as the biggest knowledge base to extract as much accurate knowledge as possible without any involvement of domain-specific human efforts
System Architecture

Search Engine    Web Corpus

KnowItAll    TextRunner

Extractor

Assessor

Database
Secret Source: Domain-Independent Rules

• The system is given a set of classes the users are interested in
  – E.g., City, Scientist, Film
• Domain-independent rules

NP1 {“,”} “such as” NPList2
NP1 {“,”} “and other” NP2
NP1 {“,”} “including” NPList2
NP1 “is a” NP2
NP1 “is the” NP2 “of” NP3
“the” NP1 “of” NP2 “is” NP3
Extractor

NP1 "such as" NPList2
& head(NP1)= plural(Class1)
& properNoun(head(each(NPList2)))
=> instanceof(Class1, head(each(NPList2)))

keywords: "plural(Class1) such as"

*The New York Observer:*

The Partnership for New York City and PricewaterhouseCoopers have a new study out measuring global cities, and New York scores high, leading many categories with cities such as London, Tokyo and Paris.

- For each pattern and each class
  - Create a search engine query
  - Pose the query to multiple search engines and collect all result documents
  - Apply the pattern to those documents and extract tuples according to the rules
**Assessor**

- Leverage the notion of PMI: Point-wise Mutual Information
  - We learned a similar concept in association rule mining: Lift = \( \frac{P(XY)}{P(X)P(Y)} \)

- For each class, KnowItAll generates a *discriminator phrase* \( D \), usually just the name of the class, then:

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

- The higher the PMI value, the more likely the instance is a true instance
  - i.e., the instance is always associated with the class
  - \( \text{Hits}(D) \) can be ignored because it is the same across all instances
Improving Recall: Rule Learning

• Similar to Snowball
• Learn new rules based on extracted tuples of high confidence
• Using those new rules to extract more tuples

<table>
<thead>
<tr>
<th>Rule</th>
<th>Correct Ex extractions</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>the cities of &lt;city&gt;</td>
<td>5215</td>
<td>0.80</td>
</tr>
<tr>
<td>headquartered in &lt;city&gt;</td>
<td>4837</td>
<td>0.79</td>
</tr>
<tr>
<td>for the city of &lt;city&gt;</td>
<td>3138</td>
<td>0.79</td>
</tr>
<tr>
<td>in the movie &lt;film&gt;</td>
<td>1841</td>
<td>0.61</td>
</tr>
<tr>
<td>&lt;film&gt; the movie starring</td>
<td>957</td>
<td>0.64</td>
</tr>
<tr>
<td>movie review of &lt;film&gt;</td>
<td>860</td>
<td>0.64</td>
</tr>
<tr>
<td>and physicist &lt;scientist&gt;</td>
<td>89</td>
<td>0.61</td>
</tr>
<tr>
<td>physicist &lt;scientist&gt;,</td>
<td>87</td>
<td>0.59</td>
</tr>
<tr>
<td>&lt;scientist&gt;, a British scientist</td>
<td>77</td>
<td>0.65</td>
</tr>
</tbody>
</table>
Improving Recall: Subclass Extraction

• Apply a subset of the domain-independent rules to extract subclasses instead of instances
  – E.g., “physicists and other scientists”
  – Distinguishing between instances and subclasses
    • Common noun (no capitalization)
    • WordNet matching
    • Similarity to the class name (e.g., biologists and microbiologists)

• Using enumeration rules to extract more subclasses after obtaining an initial set
  – E.g., “Biologists, physicists, and chemists have come together …”

• Then augment the set of domain-independent rules with those subclasses rules
  – E.g., “scientists such as …” ➔ “physicists such as …”
Improving Recall: List Extraction

• Compose search queries that contain \( k \) number of random instances from the class
• For each returned document, search for lists in the document
• Extract the unknown entries in the list as candidate instances
• Rank those instances based on
  – How many lists they appear in
  – How good are the lists they appear in (i.e., how many known instances those lists contain)
Recall Experiments

Figure 2: Instances of City at precision .90.
System Architecture

Search Engine  Web Corpus

KnowItAll  TextRunner

Extractor

Assessor

Database
TextRunner Goals

- Eliminate dependency on search engines
- Extract all kinds of relations (in addition to `instanceOf` or `isA`)
- Both are addressed by using a self-supervised learner
  - Idea: using a deep linguistic classifier to generate training examples and train a classifier over a small set of documents, and leverage shallow features to estimate trustworthiness for the entire Web.
Self-Supervised Learner

- Deep linguistic parser to generate training examples to estimate trustworthiness of relations
  - Linguistic parser: an NLP tool that converts a sentence into a syntax parse tree

Conan put every sword into the box
Self-Supervised Learner

- Deep linguistic parser to generate training examples to estimate trustworthiness of relations
  - Linguistic parser: an NLP tool that converts a sentence into a syntax parse tree
  - An entity tagger that tags any base noun phrase as a potential entity
  - For each pair of potential entities in the sentence
    - Collect deep parse tree features: dependency, pronoun, etc.
  - Label a pair as a positive example if certain deep features are satisfied, label it as a negative example otherwise
Self-Supervised Learner

- For each positive/negative example, collect shallow parse features
  - Number of tokens between entities
  - Number of stop words
  - Etc.

- Train a trustworthiness classifier based on those features and the examples
  - Given a pair of entities and their label, produce a score as to whether their relation is likely to be true or not
Extractor and Assessor

- Play similar roles as their counterparts in KnowItAll
- The extractor extract candidate tuples
  - Step 1: raw candidate generation, essentially using a simple part-of-speech tagger to generate pairs of entities
  - Step 2: candidate classification using the classifier built earlier
- The assessor ranks the remaining candidates using a probabilistic model based on counts
  - An improvement over the PMI model
Statistics of Extracted Tuples

- **Well-formed relation**: the relation label is semantically meaningful
- **Well-formed entities**: the entities are indeed semantically meaningful to the relation label
- **Abstract vs. concrete**
  - <CEO, manages, company> vs
  - <Larry Page, manages, Google>
Summary

• KnowItAll/TextRunner is the first true web-scale information system

• Open information extraction, in contrast to community-centric information extraction (such as Cimple), represents an opposite direction of research

• Both directions are important and they are complementary in terms of the target users and the intended use of the results
Quiz + 10 min Break
Outline

• Information extraction systems
  – (Snowball)
  – Cimple
  – KnowItAll/TextRunner

• Quiz + Break (30 min)

• Some recent research
  – Enhancing Wikipedia
  – Prioritization of information extraction

• Conclusion
Bootstrapping Semantic Web with IE Techniques


• Leveraging Wikipedia to bootstrap the creation of Semantic Web
  - Addressing the chicken-and-egg problem

• Specifically
  - Infobox completion
    • Adding missing infoboxes
    • Adding missing attributes
  - Link generation
    • Adding missing links to the article
Wikipedia Infobox

Polytechnic Institute of New York University

The Polytechnic Institute of New York University (also known as Polytechnic Institute of NYU or NYU-Poly) is the second oldest private institute of technology in the United States. It was founded in 1854 in the Borough of Brooklyn in New York City, and has a distinguished history in electrical engineering, polymer chemistry, aerospace, and microwave engineering. It was also known for its outreach programs to encourage math and science education in New York elementary and high schools.

In addition to its main address at MetroTech Center in Downtown Brooklyn, the institute offers programs at other sites throughout the region, including Long Island, Westchester, and Manhattan, as well as several programs in Israel. NYU-Poly also maintains a dual degree program with Stevens University/Stevens Institute of Technology.

NYU-Poly was listed among the top 10 innovative I.T. schools by Computerworld.com, top four in the U.S. for student diversity by U.S. News & World Report and one of the best Northeastern colleges by the Princeton Review. Its Carnegie Classification is Doctorate-Granting "Research University" high research activity.

NYU-Poly has been ranked, according to the 2010 US News, among the best Engineering Graduate Schools in the nation.

Among its graduates and faculty are Nobel and Wolf Prize laureates, notable inventors, world class scientists and successful entrepreneurs.

structured data

{{infobox University
| name = Polytechnic Institute of New York University
| logo = [[File:NYU-Poly.png]]
| motto = Homo et Hominis
| mottoeng = The human being
| established = 1854
| type = [[Private school]]
| endowment = $93.8 million
| Fiscal Year 2009 Endowment = 2009 
| work = 2009 
| NACUBO-Common
| Business Officers | url = /2009_NCSE_Public_Tables
| faculty = 125
| president = [[Jerry Hult
| students = 2819
| undergrad = 1543
| city = [[Brooklyn]]
| state = [[New York|NY]]
| coord = (40.6944
| country = [[United States]]
| campus = [[United area|Urban area]]
| colors = Purple and Green
| mascot = <!!- [[File:BlueJay.png]]
| website = [http://www.poly.edu
| image = [[File:Poly logo.png]]
}}

Fighting Blue Jay

Websites

www.poly.edu


Why Wikipedia?

• Relatively clean information source with a broad coverage
  – On the general Web, spam and misleading sources are much more common and difficult to deal with
  – Covers all popular domains, unlike specialized sites like IMDB or ESPN

• Concepts are typically uniquely identified
  – Synonyms are connected through redirects
  – Homonyms are distinguished via disambiguation pages

• Structured data already present
  – Infobox
  – Categories
  – Lists
Infobox Completion

- Learn from existing infoboxes how to extract information from the article to populate incomplete infoboxes or adding infoboxes to articles
- Three main components:
  - Preprocessor (training example generation)
  - Classifier
  - Extractor
Challenges

- Infobox are generated by human
  - Incomplete & Inconsistent
  - Heterogeneity
    - Schema name heterogeneity: single conceptual category may have multiple instantiations
      - U.S. County, US County, Counties, County
    - Schema attribute heterogeneity: single conceptual attribute may have multiple names
      - Census Yr, Census Year, Census Estimate Yr

- Lists and categories have similar problems
Preprocessor

• **Schema Refinement**
  – Simple statistical approach
  – Ignores schema name heterogeneity
  – For each schema, collect the most frequent attribute names with a minimum coverage threshold (15%)

• **Training example construction**
  – Identify sentences that contain the attribute values within the infobox
  – E.g., “It has a total area of 123 square miles” matches `{{area=123 sqmi}}`
  – Several heuristics:
    • Identifier standardization
    • Leverage attribute name to break the tie

• **A bunch of heuristics, but seem to work well enough**
Classifier

• Given an article that is missing an infobox or whose infobox is incomplete
  – Step 1: decide which kind of infobox should be associated with this article → document classifier
  – Step 2: decide what attribute values can appear in the infobox → sentence classifier
• **Document classifier: two heuristics**
  – The document belong to a list with a title containing the schema name
  – The document belongs to a category with a title containing the schema name
• **Sentence classifier: machine learned**
  – Maximum entropy model with terms in the sentence as features
Extractor

- Given a sentence predicted to be containing the value of an attribute, extract that value
  - E.g., “It has a total area of 123 square miles” is predicted to contain “area”, the extractor aims to extract “123” as the attribute value

- Standard CRF classifier
  - CRF: conditional random field

- Three adjustments:
  - Negative example cleaning
  - Using the classifier decision as a feature
  - Recognize multi-valued attributes
Some Experimental Results

<table>
<thead>
<tr>
<th>Class</th>
<th>People</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pre.(%)</td>
<td>Rec.(%)</td>
</tr>
<tr>
<td>County</td>
<td>97.6</td>
<td>65.9</td>
</tr>
<tr>
<td>Airline</td>
<td>92.3</td>
<td>86.7</td>
</tr>
<tr>
<td>Actor</td>
<td>94.2</td>
<td>70.1</td>
</tr>
<tr>
<td>University</td>
<td>97.6</td>
<td>90.5</td>
</tr>
</tbody>
</table>

Table 4: Relative performance of people and KYLIN on infobox attribute extraction.
Summary

• Wikipedia has become a tremendous source of knowledge for humans to consume, however it is still hard for a machine to understand.
• This paper is a step toward making knowledge machine understandable.
• While the techniques are mostly heuristic, the problem being addressed is interesting and important.
Prioritization of Domain-Specific Web Information Extraction

Jian Huang and Cong Yu,
In AAAI, Atlanta, GA, 2010.
Motivation

• **Vertical property: Yahoo! Local**
  – Maintains listings of local businesses (schools, restaurants, dentists, etc.)
  – Provides structured information (address, phone, etc.)
  – Yahoo spend lots of money in licensing fees to obtain those listings from third parties

**Goal:** Using information extraction to automatically obtain local business listings from the Web
Our Solution: Prioritization

• Naïve approach
  – Perform extraction on all the pages on the Web

• Classifier approach
  – Build a filter that removes irrelevant pages first
  – Not applicable to many domains where it is difficult to design a good filter

Given a large corpus such as the Web and a given domain such as restaurants, determine the order of the pages to be extracted, at a reasonable additional cost to the actual extraction, such that the value of the extraction is maximized given the available resources.
Overview and Goals

- Propose a notion of **Consensus Graph Value (CG Value)** to measure the value of the extraction results based on
  - The number of results
  - The given query workload
- Define a notion of **Page Utility** to estimate a given page’s contribution to the extraction results
- Design efficient page utility estimation and page scheduling algorithms
Criteria for Selecting Pages

• **Observation**: not all pages are created equal!

• **Relevance**: is this page about the topic that we are interested in?

• **Importance**: can *important* entities and relations be extracted from the page?

• **Extractability**: can our extraction system extract from this page?

• **Novelty**: will new information be obtained by extracting this page?

• **Density**: how much information can we extract per unit of computation?

• **Corroboration**: will we be able to resolve existing conflicts or increase confidence based on what we will extract from this page?
Consensus Graph (CG)

- Our logical data model for storing extraction results
  - **E**: A set of typed entities
  - **A**: A set of atomic values
  - **R**: $E \times (E \cup A) \times \text{String}$, the set of labeled relations
- Each entity can have multiple relations and each relation can have multiple values
CG Value

• **Intuition:** a CG has a higher value if it *satisfies* more user queries
  – Matching + Returning previously unknown information

  Q1: joe’s shanghai restaurant
  Q2: joe’s shanghai manhattan
  Q3: french restaurants new york
  Q4: chinese restaurants manhattan

• **Coverage semantics**
  – Instance query
    • A few matching results, each one is important
  – Category query
    • Many matching results, after a certain threshold, additional results are less important
The CG Value Formula

\[ V_{CG}^{QW} = \sum_{q \in QW} V(CG, q) = \sum_{q \in QW} \sum_{i=1}^{n_q} \alpha^{p_i} V(e_i, q) \]  

where  \( \alpha > 1, p_i = \begin{cases} 0 & \text{q is an instance query} \\ 1 - i & \text{q is a category query} \end{cases} \)

- \( i \): the \( i \)-th entity matching the query
- \( \alpha \): decay factor for category queries
- Note 1: entities matching more queries or more frequent queries contribute more to the value
- Note 2: for category queries, the contributions of matching entities are decreased according to their rank positions
The CG Value Formula

\[ \mathcal{V}(e_i, q) = \sum_{j=1}^{J} \mathcal{DG}(r_{(i,j)}, q) \]

\[ \mathcal{DG}(r_{(i,j)}, q) = \sum_{k=1}^{K} \beta^{1-k} c, \quad r_{(i,j)}^k \notin q \]

- \( \beta \): diminishing gain factor
- \( c \): constant value unit (set to 1)
- Note 1: Using diminishing gain addresses the problem of redundant information without using expensive data integration techniques
- Note 2: CG Value can be customized to express other more primitive measures for extraction results, such as \# of entities
Page Utility

• Given a page \((p)\), we measure the potential contribution of the page to the CG.

• Two main factors:
  – What can be extracted from the page
  – What’s already there in the current CG

\[
U(p) = E[V_{CG+ext(p)}] \cdot Pr(p) + V_{CG} \cdot (1 - Pr(p)) - V_{CG} \\
= Pr(p) \cdot (E[V_{CG+ext(p)}] - V_{CG})
\]
Estimating Page Utility

• **Pr(p)**
  - Pr(p is relevant): estimated using a light-weight extractor
  - Pr(p is extractible | p is relevant): estimated based on accumulated knowledge about the extractor

• **E[V_{CG+ext(p)}]**
  - Much harder to estimate
  - The set of possible entities within p are estimated from the page URL (metadata) or page header and footer (content) without examining bulk of the main content
  - The redundancy is estimated by checking the possible entities against the current set of extracted entities
  - The set of non-redundant entities are then compared against the query workload to estimate their total contribution
Batch-Oriented Architecture

- **s**: sample size
- **b**: batch size
- **k**: re-estimation size
Comparison with Random Strategy
Comparison with Classifier Approach

![Graph showing CG Value vs. Pages for prioritization, score 0.2, score 0.8, and random methods.](image)
Contributions

• Proposed CG Value, which provides a more comprehensive metric for measuring the value of extraction results
• Established Prioritization as a flexible approach for maximizing extraction benefits given any constraint on the computational resources
Summary of Information Extraction

• **Basic techniques**
  – Name entity recognizer
  – Wrapper
  – Natural language processor

• **System**
  – Domain-centric vs Open
  – Human scalable
  – Performance scalable
    • Cost based ptimization
    • Prioritization
Questions?