What is “Information Extraction”

As a task: Filling slots in a database from sub-segments of text.

What is “Information Extraction”

As a family of techniques:

Information Extraction = segmentation + classification + association + clustering.

Landscape of IE Tasks (2/4): Pattern Scope

- Web site specific
- Genre specific
- Wide, non-specific

Amazon Book Pages

Resumes

University Names

Landscape of IE Tasks (1/4): Pattern Feature Domain

- Text paragraphs without formatting
- Grammatical sentences and some formatting & links
- Non-grammatical snippets, rich formatting & links
- Tables

October 14, 2002, 4:00 a.m. PT

For years, Microsoft Corporation CEO Bill Gates railed against the economic philosophy of open-source software with Orwellian fervor, denouncing its communal licensing as a "cancer" that stifled technological innovation.

Today, Microsoft claims to "love" the open-source concept, by which software code is made public to encourage improvement and development by outside programmers. Gates himself says Microsoft will gladly disclose its crown jewels—the coveted code behind the Windows operating system—to select customers.

"We can be open source. We love the concept of shared source," said Bill Veghte, a Microsoft VP. "That's a super-important shift for us in terms of code access."

Richard Stallman, founder of the Free Software Foundation, countered saying...
Landscape of IE Tasks (3/4):

**Pattern Complexity**

E.g. word patterns:

- **Closed set**
  - U.S. states
  - Alabama

- **Regular set**
  - U.S. phone numbers
  - (413) 545-1234

- **Complex pattern**
  - U.S. postal addresses
  - University of Arkansas
  - P.O. Box 140
  - Hope, AR  71802

- **Ambiguous patterns,** needing context and many sources of evidence
  - University of Arkansas
  - P.O. Box 140
  - Hope, AR  71802

- **Person names**
  - Pawel Opalinski
  - Software Engineer at WhizBang Labs.

Jack Welch will retire as CEO of General Electric tomorrow. The top role at the Connecticut company will be filled by Jeffrey Immelt.

“Named entity” extraction

Landscape of IE Models

**Lexicons**

Classify Pre-segmented Candidates

- Abraham Lincoln was born in Kentucky.
- Abraham Lincoln was born in Kentucky.

Boundary Models

- Abraham Lincoln was born in Kentucky.

Any of these models can be used to capture words, formatting or both.

Bayesian Categorization

- Let set of categories be \( \{ c_1, c_2, \ldots, c_n \} \).
- Let \( E \) be description of an instance.
- Determine category of \( E \) by determining for each \( c_i \)
  \[ P(c_i \mid E) = \frac{P(E \mid c_i)P(c_i)}{P(E)} \]
- \( P(E) \) can be determined since categories are complete and disjoint.
  \[ \sum_{i=1}^{n} P(c_i \mid E) = \sum_{i=1}^{n} \frac{P(E \mid c_i)P(c_i)}{P(E)} = 1 \]
  \[ P(E) = \sum_{i=1}^{n} P(c_i)P(E \mid c_i) \]
Naïve Bayesian Motivation

• Problem: Too many possible instances (exponential in \( m \)) to estimate all \( P(E | c_i) \)

• If we assume features of an instance are independent given the category \( (c_i) \) (conditionally independent).

\[
P(E | c_i) = P(e_1 \land e_2 \land \cdots \land e_m | c_i) = \prod_{j=1}^{m} P(e_j | c_i)
\]

• Therefore, we then only need to know \( P(e_j | c_i) \) for each feature and category.

Recap: Naïve Bayes

• Assumption: features independent given label
• Generative Classifier
  – Model joint distribution \( p(x,y) \)
  – Inference
  – Learning: counting
  – Can we use for IE directly?

Probabilistic Graphical Models

• Nodes = Random Variables
• Directed Edges = Causal Connections
  – Associated with a CPT (conditional probability table)

Sliding Windows

Extraction by Sliding Window

E.g.
Looking for

 schlumberger 7500 Wean Hall

E.g.
Looking for

 schlumberger 7500 Wean Hall
Machine learning has evolved from obscurity in the 1970s into a vibrant and popular discipline in artificial intelligence during the 1980s and 1990s. As a result of its success and growth, machine learning is evolving into a collection of related disciplines: inductive concept acquisition, analytic learning in problem solving (e.g., analogy, explanation-based learning), learning theory (e.g., PAC learning), genetic algorithms, connectionist learning, hybrid systems, and so on.

A “Naïve Bayes” Sliding Window Model

Estimate \( \Pr(\text{LOCATION} | \text{window}) \) using Bayes rule

Try all “reasonable” windows (vary length, position)

Assume independence for length, prefix, suffix, content words

Estimate from data quantities like: \( \Pr(\text{“Place” in prefix} | \text{LOCATION}) \)

If \( \Pr(\text{“Wean Hall Rm 5409”} = \text{LOCATION}) \) is above some threshold, extract it.

Other examples of sliding window: [Baluja et al 2000]

Realistic sliding-window-classifier IE

- What windows to consider?
  - all windows containing as many tokens as the shortest example, but no more tokens than the longest example
- How to represent a classifier? It might:
  - Restrict the length of window;
  - Restrict the vocabulary or formatting used before/after/inside window;
  - Restrict the relative order of tokens, etc.
- Learning Method
  - SRV: Top-Down Rule Learning [Frätag AAAI ’98]
  - Rapier: Bottom-Up [Califf & Mooney, AAAI ’99]
Rapier – results vs. SRV

<table>
<thead>
<tr>
<th>System</th>
<th>stime</th>
<th>etime</th>
<th>loc</th>
<th>speaker</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Proc</td>
<td>Rec</td>
<td>Proc</td>
<td>Rec</td>
</tr>
<tr>
<td>Rapier</td>
<td>93.9</td>
<td>92.9</td>
<td>94.5</td>
<td>94.6</td>
</tr>
<tr>
<td>RAP-WT</td>
<td>96.5</td>
<td>93.9</td>
<td>96.8</td>
<td>96.8</td>
</tr>
<tr>
<td>RAP-W</td>
<td>96.5</td>
<td>93.9</td>
<td>96.8</td>
<td>96.8</td>
</tr>
<tr>
<td>Naïday</td>
<td>98.2</td>
<td>98.2</td>
<td>95.2</td>
<td>98.7</td>
</tr>
<tr>
<td>SRV</td>
<td>98.6</td>
<td>98.4</td>
<td>67.7</td>
<td>92.6</td>
</tr>
<tr>
<td>WHISK</td>
<td>86.2</td>
<td>100.0</td>
<td>85.6</td>
<td>87.2</td>
</tr>
<tr>
<td>WH-P</td>
<td>96.2</td>
<td>100.0</td>
<td>80.5</td>
<td>87.2</td>
</tr>
</tbody>
</table>

Rule-learning approaches to sliding-window classification: Summary

- SRV, Rapier, and WHISK [Soderland KDD '97]
  - Representations for classifiers allow restriction of the relationships between tokens, etc
  - Representations are carefully chosen subsets of even more powerful representations based on logic programming (ILP and Prolog)
  - Use of these “heavyweight” representations is complicated, but seems to pay off in results
- Can simpler representations for classifiers work?

BWI: Learning to detect boundaries

- Another formulation: learn 3 probabilistic classifiers:
  - \( \text{START}(i) = \text{Prob( position } i \text{ starts a field)} \)
  - \( \text{END}(j) = \text{Prob( position } j \text{ ends a field)} \)
  - \( \text{LEN}(k) = \text{Prob( an extracted field has length } k \text{)} \)
- Score a possible extraction \((i, j)\) by \( \text{START}(i) \times \text{END}(j) \times \text{LEN}(j-i) \)
- \( \text{LEN}(k) \) is estimated from a histogram

BWI: Learning to detect boundaries

- BWI uses boosting to find “detectors” for \( \text{START} \) and \( \text{END} \)
- Each weak detector has a \( \text{BEFORE} \) and \( \text{AFTER} \) pattern (on tokens before/after position \( i \)).
- Each “pattern” is a sequence of
  - tokens and/or
  - wildcards like: anyAlphabeticToken, anyNumber, ...
- Weak learner for “patterns” uses greedy search (+ lookahead) to repeatedly extend a pair of empty \( \text{BEFORE,AFTER} \) patterns

Problems with Sliding Windows and Boundary Finders

- Decisions in neighboring parts of the input are made independently from each other.
  - Naïve Bayes Sliding Window may predict a “seminar end time” before the “seminar start time”.
  - It is possible for two overlapping windows to both be above threshold.
  - In a Boundary-Finding system, left boundaries are laid down independently from right boundaries, and their pairing happens as a separate step.
Landscape of IE Techniques (1/1):

Models

Lexicons

Classify Pre-segmented Candidates

Sliding Window

Boundary Models

Finite State Machines

Context Free Grammars

Each model can capture words, formatting, or both

Finite State Machines

Hidden Markov Models (HMMs)

standard sequence model in genomics, speech, NLP, ...

Finite state model

Graphical model

Generates:

\[ P(\mathbf{x}, \mathbf{o}) \propto \prod_{t=1}^{T} P(\mathbf{s}_t) P(\mathbf{o}_t | \mathbf{s}_t) \]

Parameters: for all states \( S = \{s_1, s_2, \ldots \} \)

- Start state probabilities: \( P(s_1) \)
- Transition probabilities: \( P(s_t | s_{t-1}) \)
- Observation (emission) probabilities: \( P(o_t | s_t) \)

Training:

Maximize probability of training observations

Example: The Dishonest Casino

A casino has two dice:

- Fair die
  \[ P(1) = P(2) = P(3) = P(5) = P(6) = 1/6 \]
- Loaded die
  \[ P(1) = P(2) = P(3) = P(5) = 1/10 \]
  \[ P(6) = 1/2 \]

Casino player switches back-and-forth between fair and loaded die once every 20 turns

Game:

1. You bet $1
2. You roll (always with a fair die)
3. Casino player rolls (maybe with fair die, maybe with loaded die)
4. Highest number wins $2

Question # 1 – Evaluation

GIVEN
A sequence of rolls by the casino player

124552646214613613666166466163661664661636616361...

QUESTION
How likely is this sequence, given our model of how the casino works?

This is the EVALUATION problem in HMMs

Question # 2 – Decoding

GIVEN
A sequence of rolls by the casino player

124552646214613613666166466163661664661636616361...

QUESTION
What portion of the sequence was generated with the fair die, and what portion with the loaded die?

This is the DECODING question in HMMs
Question # 3 – Learning

GIVEN
A sequence of rolls by the casino player
124552646146136616664661636616361651…

QUESTION
How “loaded” is the loaded die? How “fair” is the fair die? How often does the casino player change from fair to loaded, and back?

This is the LEARNING question in HMMs

The dishonest casino model

What’s this have to do with Info Extraction?

What’s this have to do with Info Extraction?

IE with Hidden Markov Models

Given a sequence of observations:

Yesterday Pedro Domingos spoke this example sentence.

and a trained HMM:

Find the most likely state sequence: (Viterbi) \( \arg \max_x P(\hat{\alpha}, \hat{\beta}) \)

Any words said to be generated by the designated “person name” state extract as a person name:

Person name: Pedro Domingos

IE with Hidden Markov Models

For sparse extraction tasks:

- Separate HMM for each type of target
- Each HMM should
  – Model entire document
  – Consist of target and non-target states
  – Not necessarily fully connected
Or ... Combined HMM
• Example – Research Paper Headers

HMM Example: “Nymble”
Task: Named Entity Extraction
[Freitag and McCallum ’99]

Other examples of shrinkage for HMMs in IE: [Freitag and McCallum ’99]

Task: Named Entity Extraction
Train on ~500k words of news wire text.
Results:

<table>
<thead>
<tr>
<th>Case</th>
<th>Language</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixed</td>
<td>English</td>
<td>93%</td>
</tr>
<tr>
<td>Upper</td>
<td>English</td>
<td>91%</td>
</tr>
<tr>
<td>Mixed</td>
<td>Spanish</td>
<td>90%</td>
</tr>
</tbody>
</table>

Person
Org
Other
(Five other name classes)

Person
Org
Other
(Five other name classes)

Finite State Model

Question #1 – Evaluation
GIVEN
A sequence of observations \( x_1 x_2 x_3 x_4 \ldots x_N \)
A trained HMM
\( \theta = (\pi, A, \lambda) \)

QUESTION
How likely is this sequence, given our HMM ?
\( P(x, \theta) \)

Why do we care?
Need it for learning to choose among competing models!

Question #2 - Decoding
GIVEN
A sequence of observations \( x_1 x_2 x_3 x_4 \ldots x_N \)
A trained HMM
\( \theta = (\pi, A, \lambda) \)

QUESTION
How do we choose the corresponding parse (state sequence) \( y_1 y_2 \ldots y_N \), which “best” explains \( x_1 x_2 x_3 x_4 \ldots x_N \) ?

There are several reasonable optimality criteria:
- single optimal sequence, average statistics for individual states, ...

Slide by Okan Basoglu
A parse of a sequence
Given a sequence $x = x_1 \ldots x_N$, a parse of $o$ is a sequence of states $y = y_1 \ldots y_N$

Given a sequence $x = x_1 \ldots x_N$, a parse of $o$ is a sequence of states $y = y_1 \ldots y_N$

Question #3 - Learning
GIVEN
A sequence of observations $x_1 x_2 x_3 x_4 \ldots x_N$

QUESTION
How do we learn the model parameters $\theta$ which maximize $P(x, \theta)$?

Three Questions
- Evaluation
  - Forward algorithm
  - (Could also go other direction)
- Decoding
  - Viterbi algorithm
- Learning
  - Baum-Welch Algorithm (aka “forward-backward”)
  - A kind of EM (expectation maximization)

A Solution to #1: Evaluation
Given observations $x = x_1 \ldots x_N$ and HMM $\theta$, what is $p(x)$?

Enumerate every possible state sequence $y = y_1 \ldots y_N$

Probability of $x$ and given particular $y$

Probability of particular $y$

Summing over all possible state sequences we get

$N \times 2^T$ multiplications

For small HMMs $T=10, N=10$

there are $10^{10}$ billion sequences!

Solution to #1: Evaluation
Use Dynamic Programming:
Define forward variable

$\alpha_t(i) = P(x_1 x_2 \ldots x_t, y_t = S_i)$

2T multiplications per sequence

Forward Variable

$\alpha_t(i) = P(x_1 x_2 \ldots x_t, y_t = S_i)$

Prob - that the state at time $t$ was value $S_i$ and
- the partial obs sequence $x_1 x_2 \ldots x_t$ has been seen

- the state is $S_i$
- the partial observation sequence $x_1 x_2 \ldots x_t$ has been emitted
The Forward Algorithm

Forward Variable $\alpha_t(i)$

\[ \alpha_t(i) = P(x_1 x_2 \ldots x_t, y_t = S_i) \]

- prob that the state at t was value $S_i$, and
- the partial obs sequence $x_1 \ldots x_t$ has been seen

The Backward Algorithm

Backward Variable $\beta_t(i)$

\[ \beta_t(i) = P(y_t \ldots y_N | x_1 \ldots x_t, y_t = S_i) \]

- prob that at time t
  - the state is $y_t = S_i$
  - the partial observation sequence $x_1 \ldots x_t$ has been omitted

Solution to #1: Evaluation

\[ p(x) = \sum_{y} \prod_{t=1}^{T} p(y_t | y_{t-1}) p(x_t | y_t) \]

- Cache and reuse inner sums
- Define forward variables

\[ \alpha_t(i) := P(x_1 x_2 \ldots x_t, y_t = S_i) \]

Time: $O(KN)$

Space: $O(KN)$

$K = |S|$, #states

$N$ length of sequence

The Forward Algorithm

INITIALIZATION

\[ \alpha_0(i) = p(y_0 = S_i | x_1) \]

INDUCTION

\[ \alpha_t(i) = \sum_{j \in S} \alpha_{t-1}(j) p(y_t = S_i | x_t = y_t) \]

TERMINATION

\[ p(x) = \sum_{i \in S} \alpha_N(i) \]

Time: $O(KN)$

Space: $O(KN)$

The Backward Algorithm

INITIALIZATION

\[ \beta_T(i) = 1 \]

INDUCTION

\[ \beta_t(i) = \sum_{j \in S} \beta_{t+1}(j) p(y_{t+1} = S_j | x_t = y_t) \]

TERMINATION

\[ p(x) = \sum_{i \in S} \beta_0(i) p(x_1 | y_0) \alpha_0(i) \]

Time: $O(KN)$

Space: $O(KN)$

$K = |S|$, #states

$N$ length of sequence
Three Questions

• Evaluation
  – Forward algorithm
  – (Could also go other direction)
• Decoding
  – Viterbi algorithm
• Learning
  – Baum-Welch Algorithm (aka "forward-backward")
  – A kind of EM (expectation maximization)

Solution to #2 - Decoding

Given \( x = x_1 \ldots x_N \) and HMM \( \theta \), what is "best" parse \( y_1 \ldots y_N \)?

Several optimal solutions

• 1. States which are individually most likely
• 2. Single best state sequence

We want to find sequence \( y_1 \ldots y_N \)

such that \( P( x, y ) \) is maximized

\[
   y^* = \arg\max_y P( x, y )
\]

Again, we can use dynamic programming!

The Viterbi Algorithm

```
DEFINE
   \( \alpha_i(0) \) = \( \frac{P(x_1) \theta_{i1}}{} \)

INITIALIZATION

INDUCTION
   \( \alpha_i(t) = \max_j \{ \alpha_j(t-1) \cdot \theta_{ji} \} \) \( x_t \)

TERMINATION
   \( \delta_i(T) = \max_j \{ \alpha_j(T-1) \cdot \theta_{ji} \} \) \( x_T \)
```

Remember: \( \delta_i(j) \) = probability of most likely state seq ending with state \( S_i \)

Time: \( O(KT) \) – Linear in length of sequence

Space: \( O(KT) \)

Need new slide!
Three Questions

- Evaluation
  - Forward algorithm
  - (Could also go other direction)
- Decoding
  - Viterbi algorithm
- Learning
  - Baum-Welch Algorithm (aka "forward-backward")
  - A kind of EM (expectation maximization)

Solution to #3 - Learning
Given $x_1 \ldots x_N$, how do we learn $\theta = (\ldots)$ to maximize $P(x)$?

- Unfortunately, there is no known way to analytically find a global maximum $\theta^*$ such that
  $\theta^* = \text{arg max } P(o | \theta)$

- But it is possible to find a local maximum; given an initial model $\theta$, we can always find a model $\theta'$ such that
  $P(o | \theta') \geq P(o | \theta)$

Chicken & Egg Problem

- If we knew the actual sequence of states
  - It would be easy to learn transition and emission probabilities
  - But we can't observe states, so we don't!

- If we knew transition & emission probabilities
  - Then it'd be easy to estimate the sequence of states (Viterbi)
  - But we don't know them!

Simplest Version

- Mixture of two distributions

  \[ \text{Distribution 1} \quad \text{Distribution 2} \]

- Know % = 5
- Just need mean of each distribution

Input Looks Like

We Want to Predict
Chicken & Egg

Note that coloring instances would be easy if we knew Gausians….

And finding the Gausians would be easy if we knew the coloring.

Expectation Maximization (EM)

- Pretend we do know the parameters
  - Initialize randomly: set $\theta_1=?; \ \theta_2=?$

- [E step] Compute probability of instance having each possible value of the hidden variable

- [M step] Treating each instance as fractionally having both values compute the new parameter values
ML Mean of Single Gaussian

\[ U_{ml} = \arg\min_u \sum_i (x_i - u)^2 \]

Expectation Maximization (EM)

- **[E step]** Compute probability of instance having each possible value of the hidden variable

<table>
<thead>
<tr>
<th>0.01</th>
<th>0.03</th>
<th>0.05</th>
<th>0.07</th>
<th>0.09</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **[M step]** Treating each instance as fractionally having both values compute the new parameter values

Expectation Maximization (EM)

- **[E step]** Compute probability of instance having each possible value of the hidden variable

EM for HMMs

- **[E step]** Compute probability of instance having each possible value of the hidden variable
  - Compute the forward and backward probabilities for given model parameters and our observations

- **[M step]** Treating each instance as fractionally having both values compute the new parameter values
  - Re-estimate the model parameters
  - Simple Counting
Summary - Learning

- Use hill-climbing
  - Called the forward-backward (or Baum/Welch) algorithm

- Idea
  - Use an initial parameter instantiation
  - Loop
    - Compute the forward and backward probabilities for given model parameters and our observations
    - Re-estimate the parameters
    - Until estimates don’t change much

IE Resources

- Data
  - Linguistic Data Consortium (LDC)
    - Penn Treebank, Named Entities, Relations, etc.
    - http://www.bioinf.wisc.edu/~craven/ie
    - http://www.cs.umass.edu/~mccallum/mallet

- Code

- Both
  - http://www.cs.upenn.edu/~adwait/penntools.html

References