Evaluating IE Accuracy  [A common way]

- Always evaluate performance on independent, manually-annotated test data not used during system development!
- Measure for each test document:
  - Total number of correct extractions in the solution template: \( N \)
  - Total number of slot/value pairs extracted by the system: \( E \)
  - Number of extracted slot/value pairs that are correct (i.e. in the solution template): \( C \)
- Compute average value of metrics adapted from IR:
  - Recall = \( C/N \)
  - Precision = \( C/E \)
  - F-Measure = Harmonic mean of recall and precision

Variants: partial match, etc.

Good Basic IE References


Information Extraction

References


Variants: partial match, etc.

Summary and prelude

- We looked at the “fragment extraction” task. Future?
- Top-down semantic constraints (as well as syntax)?
- Unified framework for extraction from regular & natural text? (BW1 is one tiny step; Webfoot [Soderland, 97] is another.)
- Beyond fragment extraction:
  - Anaphora resolution, discourse processing, ...
- Fragment extraction is good enough for many Web information services!
- Applications: What exactly is IE good for?
  - Is there a use for today’s “50%” results?
  - Palmtop devices? – IE is valuable if screen is small
- Today
  - Learning methods for information extraction
Learning for IE

- Writing accurate patterns for each slot for each domain (e.g. each web site) requires laborious software engineering.
- Alternative is to use machine learning:
  - Build a training set of documents paired with human-produced filled extraction templates.
  - Learn extraction patterns for each slot using an appropriate machine learning algorithm.

Rapier [Califf & Mooney, AAAI-99]

- Rapier learns three regex-style patterns for each slot:
  - Pre-filler pattern
  - Filler pattern
  - Post-filler pattern

- One of several recent trainable IE systems that incorporate linguistic constraints. (See also SIFT [Miller et al, MUC-7]; SRV [Frétag, AAAI-98]; Whisk [Soderland, MLJ-99].)

```
"...sold $11M for the company..."
"...paid Honeywell an undisclosed amount..."
"...paid Honeywell an undisclosed price..."
```

Pre-filler: Filler: Post-filler:
1) tag: {nn,mnp} 1) word: undisclosed 1) sem: price
2) list: length 2 (tag: jj)
RAPIER rules for extracting "transaction price"

Part-of-speech tags & Semantic classes

- Part of speech: syntactic role of a specific word
  - noun (nn), proper noun (nnp), adjective (jj), adverb (rb),
    determiner (dt), verb (vb), "", "", ...
  - NLP: Well-known algorithms for automatically assigning POS tags to English, French, Japanese, ...
    (->95% accuracy)
- Semantic Classes: Synonyms or other related words
  - "Price" class: price, cost, amount, ...
  - "Month" class: January, February, March, ..., December
  - "US State" class: Alaska, Alabama, ..., Washington, Wyoming
  - WordNet: large on-line thesaurus containing (among other things) semantic classes

Rapier Rules: Details

- Rapier rule :=
  - pre-filler pattern
  - filler pattern
  - post-filler pattern
  - pattern := subpattern +

- subpattern := constraint +
- constraint :=
  - Word = exact word that must be present
  - Tag - matched word must have given POS tag
  - Class - semantic class of matched word
  - Can specify disjunction with "[...]
  - List length N - between 0 and N words satisfying other constraints

```

```
Rapier’s Learning Algorithm

- **Input**: set of training examples (list of documents annotated with ‘extract this substring’)
- **Output**: set of rules

- **Init**: Rules = a rule that exactly matches each training example
- **Grow**
  - Select N examples randomly and generate the K most-accurate maximally general filler-only rules (prefixfiller = postfiller = ‘true’).
  - Repeat For N = 1, 2, 3, ...
  - Try to improve K best rules by adding N context words of prefixfiller or postfiller context
- **Keep**: Rules = Rules ∪ the best of the K rules - subsumed rules

Learning example (one iteration)

Statistical generative models

- Previous discussion examined systems that use explicit extraction patterns/rules
- Hidden Markov Models are a powerful alternative based on statistical token sequence generation models rather than explicit extraction patterns.

- **Pros**:
  - Well understood underlying statistical model makes it easy to use wide range of tools from statistical decision theory
  - Portable, broad coverage, robust, good recall
  - Cons:
  - Range of features and patterns usable may be limited
  - Not necessarily as good for complex multi-slot patterns

Name Extraction via HMMs

Applying HMMs to IE

- **Document** ⇒ generated by a stochastic process modelled by an HMM
- **Token** ⇒ word
- **State** ⇒ ‘reason/explanation’ for a given token
  - ‘Background’ state emits tokens like ‘the’, ‘said’, ...
  - ‘Money’ state emits tokens like ‘million’, ‘euro’, ...
  - ‘Organization’ state emits tokens like ‘university’, ‘company’, ...
- **Extraction**: via the Viterbi algorithm, a dynamic programming technique for efficiently computing the most likely sequence of states that generated a document.

HMM formalism

HMM = probabilistic FSA

- **HMM** = states \( s_1, s_2, \ldots \) (special start state \( s_1 \), special end state \( s_2 \))
- **Token alphabet** \( a_1, a_2, \ldots \)
- **State transition probs** \( P(s_i | s_j) \)
- **Token emission probs** \( P(a_k | s_j) \)

Widely used in many language processing tasks, e.g., speech recognition [Lee, 1980], POS tagging [Kupiec, 1992], topic detection [Yamron et al, 1998]
Learning HMMs

- **Good news**: If training data tokens are tagged with their generating states, then simple frequency ratios are a maximum-likelihood estimate of transition/emission probabilities. Easy. (Use smoothing to avoid zero probs for emissions/transition absent in the training data.)
- **Great news**: Baum-Welch algorithm trains an HMM using partially labeled or unlabeled training data.
- **Bad news**: How many states should the HMM contain? How are transitions constrained?
  - Only semi-good answers to finding answer automatically
  - Insufficiently expressive: Unable to model important distinctions (long distance correlations, other features)
  - Overly expressive: Sparse training data, overfitting

What is an HMM?

- Graphical Model Representation: Variables by time
- Circles indicate states
- Arrows indicate probabilistic dependencies between states

What is an HMM?

- Green circles are hidden states
- Dependent only on the previous state: Markov process
- “The past is independent of the future given the present.”

What is an HMM?

- Purple nodes are observed states
- Dependent only on their corresponding hidden state

Trained on 2 million words of BibTeX data from the Web
HMM Formalism

- \( \{S, K, \Pi, A, B\} \)
- \( S = \{s_1, ... , s_N\} \) are the values for the hidden states
- \( K = \{k_1, ... , k_M\} \) are the values for the observations

\[
\Pi = \{\pi_i\}
\]

\[
A = \{a_{ij}\}
\]

\[
B = \{b_{jk}\}
\]

Inference for an HMM

- Compute the probability of a given observation sequence
- Given an observation sequence, compute the most likely hidden state sequence
- Given an observation sequence and set of possible models, which model most closely fits the data?

Sequence Probability

\[
P(O \mid X, \mu) = \prod_{t=1}^{T} b_{s_t, y_t} \prod_{i=1}^{T} a_{s_{t-1}, s_t} P(X \mid \mu)
\]

\[
P(O \mid \mu) = \sum_{X} P(O \mid X, \mu) P(X \mid \mu)
\]
Sequence probability

- Special structure gives us an efficient solution using dynamic programming.
- Intuition: Probability of the first $t$ observations is the same for all possible $t + 1$ length state sequences.
- Define: $\alpha_i(t) = P(o_1...o_t, x_t = i | \mu)$

Forward Procedure

- $\alpha_i(t+1) = P(o_1...o_{t+1}, x_{t+1} = j)$
- $= P(o_1...o_{t+1} | x_{t+1} = j)P(x_{t+1} = j)$
- $= P(o_1...o_{t} | x_{t} = i)P(o_{t+1} | x_{t+1} = j)P(x_{t+1} = j)$
- $= P(o_1...o_{t}, x_{t+1} = j)P(o_{t+1} | x_{t+1} = j)$

Backward Procedure

- $\beta_i(T + 1) = 1$
- $\beta_i(t) = P(o_{t+1}...o_T | x_t = i)$
- $\beta_i(t) = \sum_{j=1}^{N} \alpha_j(t) a_{ji} \beta_j(t+1)$

Sequence probability

- $P(O | \mu) = \sum_{i=1}^{N} \alpha_i(T)$ Forward Procedure
- $P(O | \mu) = \sum_{i=1}^{N} \pi_i \beta_i(1)$ Backward Procedure
- $P(O | \mu) = \sum_{i=1}^{N} \alpha_i(t) \beta_i(t)$ Combination

Best State Sequence

- Find the state sequence that best explains the observations
- Viterbi algorithm (1967)
- $\arg \max_{X} P(X | O)$
### Viterbi Algorithm

\[ \delta_j(t) = \max_{x_i} P(x_i \ldots x_{t-1}, o_i \ldots o_{t+1}, x_t = j, o_t) \]

The state sequence which maximizes the probability of seeing the observations to time \( t-1 \), landing in state \( j \), and seeing the observation at time \( t \).

\[ \delta_j(t+1) = \max_j \delta_j(t) a_{ji} b_j(o_{t+1}) \]

Recursive Computation

\[ \hat{X}_T = \arg \max \hat{\delta}_j(T) \]

\[ \hat{X}_t = \psi_j \quad (t+1) \]

Compute the most likely state sequence by working backwards

\[ P(\hat{X}) = \arg \max_j \hat{\delta}_j(T) \]

### Learning = Parameter Estimation

- Given an observation sequence, find the model that is most likely to produce that sequence.
- No analytic method, so:
- Given a model and observation sequence, update the model parameters to better fit the observations.

### Parameter Estimation: Baum-Welch or Forward-Backward

\[ p_i(i, j) = \frac{\alpha_i(t) a_{ji} p_{ji} \beta_j(t+1)}{\sum_{o_{t+1}} \alpha_i(t) \beta_i(t+1)} \]

Probability of traversing an arc

\[ \gamma_i(t) = \sum_{j=1 \ldots N} p_i(i, j) \]

Probability of being in state \( i \)

\[ \hat{a}_{ji} = \frac{\sum_{t=1}^{T-1} p_i(i, j)}{\sum_{t=1}^{T-1} \gamma_i(t)} \]

\[ \hat{b}_j = \frac{\sum_{t=1}^{T-1} o_j \gamma_i(t)}{\sum_{t=1}^{T-1} \gamma_i(t)} \]

Now we can compute the new estimates of the model parameters.
Is it that easy?

- As often with text, the biggest problem is the sparseness of observations (words).
- Need to use many techniques to do it well
  - Smoothing (as in NB) to give suitable nonzero probability to unseen
  - Feature decomposition (capitalized?, number?, etc.) gives a better estimate
  - Shrinkage allows pooling of estimates over multiple states of same type (e.g., prefix states)
- Well designed (or learned) HMM topology

HMM example

“Seminar announcements” task

- PostedBy: Colin S Osborn on 15-Apr-95 at 15:11 from CMU.EDU
- Dates: 17-Apr-95
- Time: 7:00 PM
- Topic: Re: entrepreneurship speaker
- Abstract:

  hello again
  to reiterate
  there will be a speaker on the law and startup business
  this monday evening the 17th
  it will be at
  7pm
  in
  room 261 of GSIA
  in the new building, ie
  upstairs.
  please attend if you have any interest in starting your own
  business or
  are even curious.

  Colin

HMM example, continued

Fixed topology that captures limited context: 4 “prefix” states before and 4 “suffix” after target state.

Learning HMM structure (Seymore et al. 1999)

1. start with maximally-specific HMM (one state per observed word):
2. repeat
   (a) merge adjacent identical states
   (b) eliminate redundant fan-out/in
3. until obtain good tradeoff between HMM accuracy and complexity

References