Extracting Rich Event Structure from Text Models and Evaluations

Day 2: Generative Models

Nate Chambers
US Naval Academy
Previously: clustering-based
Clustering: potential problems

• Too many stages
• Too many parameters
• Too many hacky combinations
• Too many training documents
Benefits of a Probabilistic Model

1. Uses a more formal clustering algorithm
2. Adds features into the model in a modular way
3. Easier to generalize to different domains
4. Distributions rather than hard cutoffs
Random variables

• An event or slot can be represented by a random variable with a specific value.

\[
\begin{align*}
\text{CONVICTION} & \quad 0.1 \\
\text{DENIAL} & \quad 0.2 \\
\text{REPORTING} & \quad 0.3 \\
\text{TESTIFY} & \quad 0.1 \\
\text{CHARGE} & \quad 0.3
\end{align*}
\]

• The text is modeled as observed random variables.
  – e.g., \( P(w = \text{exonerated}|E = \text{CONVICTION}) = 0.45 \)
Learning event structure

• Learning means to learn the *probability distributions* that define these random variables

• Suppose we see the verb *deny* and the noun *accusation*, and we want to learn a distribution over possible events
  
  – Useful distribution:
    
    CONVICTION: 0.05  
    PLEAD: 0.85  
    CHARGE: 0.1

  – Bad distribution:
    CONVICTION: 0.33  
    PLEAD: 0.33  
    CHARGE: 0.33

• In practice, we won't have labels like CONVICTION, PLEAD, and CHARGE. Learning is unsupervised. They are just numerical identifiers.
Graphical models

- Nodes: random variables
- Edges: conditional dependencies

Thus, we visually describe the features and dependencies of what we are modeling in a compact way.
Catching the drift

• Probabilistic content model

We decide on the current topic depending on the previous topic, and on the words in the current sentence

• Sentences are represented as bigrams that are emitted based on the value of that sentence's topic random variable

(Barzilay and Lee, 2004)
Algorithm for unsupervised learning

- **Expectation maximization** algorithm

- General intuition:
  - Initialize a random model
  - Repeat the following steps for a while:
    - Guess the values of the topic variables
    - Get a better model based on the current guesses

- This process eventually leads to a better clustering of the data (i.e., better guesses) that respects the features and dependencies of the model.
Models for Event Schemas

- Event words are observed
- Syntactic arguments are observed

- Each event word has a latent event variable.
- Each argument has a latent slot (or role) variable.

Goal: learn distribution of events and slots
- Beyond the above, research differs in how to model the conditional dependencies.

The judge convicted
Model 1: ProFinder

Coherent probabilistic model similar to an HMM
  – Structure adapted to model frame components
  – Standard inference algorithms (e.g. Inside-outside, Viterbi)

• **Base** model
  a. Latent variables: Frames, events, and slots
  b. Observed variables: Event heads and arguments

• Extend with linguistics-based **refinements**
  a. Background frame
  b. Argument dependencies

• Split-merge **learning**

(Cheung and Poon, 2013)
ProFinder: graphical model

Transitions model discourse structure

Latent variables model frame components.

Arguments

Latent variables model frame components.

Emissions model observed text.

Linguistic text-based refinements

Background

Frame

Event

Event head

Arguments

Emissions model observed text.

Linguistic text-based refinements
Base: Frames, Events, and Slots

- One frame and event per clause
  - Extract from parse tree
- $N$ frame states
  - Each state $F$ comprises a unique set of events $E_F$ and slots $S_F$
- Emissions depend on:
  - Event $\rightarrow$ event head (verb or event noun)
  - Slot $\rightarrow$ NP head

Individuals threatened the investigators.
Base: Transitions

- Frame and event transitions modeled
  - Captures discourse structure
  - First-order Markov assumption
- Constraints:
  - No frame state transition within sentence
  - Event transitions respect frame assignments
Refinement: Background Frame

• Genre-specific words “interrupt” the discourse.
• e.g. **Attribution**
  – *Police reported that ...*
  – *They said ...*
• $B_i \in \{\text{BKG, CONT}\}$
  – If CONT, generate as usual
  – If BKG, generate from reserved distributions
• Preserve frame and event if BKG
  – i.e., $P(F_{i+1}|F_i, B_{i+1}) = 1(F_{i+1} = F_i)$ if $B_{i+1} = \text{BKG}$
  – Maintains discourse continuity
Refinement: Arg. Dependencies

- Syntactic relation is indicative of a slot:
  - \textit{bomb>nsubj} is \textbf{Perpetrator}
- Condition slots on relation from parse tree
- Generate the (event head, relation) pair as an emission

\begin{align*}
F_i & \rightarrow E_i \\
S_{i1} & \rightarrow a_{i1}, dep_{i1}, e_i, a_{i2}, dep_{i2} \\
& \quad \text{with } S_{i1} \rightarrow \text{Individuals} \\
& \quad \text{threatened the investigators.}
\end{align*}
Example

\[ F_1 = \text{Kidnapping} \]
\[ E_1 = \text{Kidnap} \]
\[ B_1 = \text{BKG} \]
\[ S_{11} = \text{Authorities} \]
\[ S_{21} = \text{Victim} \]
\[ S_{22} = \text{Perpetrator} \]

The command reported peasants were kidnapped by terrorists.
Learning: EM

• Choose a number of frames to learn (k)
• Give each frame one event type and two slots
  – Algorithm will grow these in number later

• Forward-backward algorithm
• Sampling to learn the event/slot distributions
• Split/Merge to grow variables
Learning: HMM parameters

• Forward-backward algorithm
  – Reduce all Frame/Event/Background combinations into single states
  – Then run forward-background to learn the parameters

• Viterbi algorithm selects most probable sequence

![Diagram of HMM parameters](image)
Learning: HMM parameters

• After forward-backward and viterbi assignment:
  • State expansion
    – Convert each $U_i$ back into its $B_i$, $F_i$, $E_i$ combination
    – $P(F=1 | F=9) = \text{Sum over all } P(U_j | U_{j-1}) \text{ s.t. } U_j=1 \text{ and } F_{i-1}=9$
    – Similarly for $E$ and $B$ values

• You now have new distributions for your 3 latent variables
• Run gibbs sampling for the observed conditionals:
Learning: HMM parameters

• You now have new distributions for your 3 latent variables (B, F, E)
• Run gibbs sampling for the remaining conditionals:
Learning: Remaining parameters

- EM for emission parameters
- Sample the latent slot variables with the observed words
- Alternate with forward-backward
Learning: Split-Merge

• Adapted from syntactic parsing (Petrov et al., 2006)
• Dynamically learn number of events and slots, $|E_F|, |S_F|$  
• Coarse-to-fine refinement of categories:  
  – e.g. persons vs. places $\rightarrow$ perpetrators vs. victims  
• Helps navigate local optima in learning  
• Repeat for several cycles:  
  1. EM iterations  
  2. Split each event and slot state  
  3. Merge states if likelihood not increased
HMM Learning Results

- MUC-4 corpus
- 1300 Latin America news articles

<table>
<thead>
<tr>
<th>Event: Attack</th>
<th>Event: Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>report, participate, kidnap, kill, release</td>
<td>hold, meeting, talk, discuss, investigate</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Slot: Perpetrator</th>
<th>Slot: Victim</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERSON/ORG</td>
<td>PERSON/ORG</td>
</tr>
</tbody>
</table>

| Words: guerrilla, police, source, person, group |
| Caseframes: report>nsubj, kidnap>nsubj, kill>nsubj, participate>nsubj, release>nsubj |

| Words: people, priest, leader, member, judge |
| Caseframes: kill>dobj, murder>dobj, release>dobj, report>dobj, kidnap>dobj |

NOTE!
The names for the events and slots were manually added. The model does not learn these. Future research idea?
HMM IE Performance?

- MUC-4 corpus
- Can evaluate how well the learned frames extract manually annotated documents.

- More on this later.
Extension: Vector Emissions

• Add word vectors as observed variables.

Cheung and Penn, 2013. Probabilistic Domain Modelling With Contextualized Distributional Semantic Vectors. ACL.
Extension: Vector Emissions

- Evaluated on TAC and a summarization task
- More about evaluations later
Model 2: Entity-Driven

- Generative model
- Represent a document as a bag of entities
- Label each entity with a latent **entity role**

(Chambers, 2013)
Why entities?

The oil stopped gushing from **BP’s ruptured well** in the Gulf of Mexico when **it** was capped on July 15 and engineers have since been working to permanently plug **it**. **The damaged Macondo well** has spewed about 4.9m barrels of oil into the gulf after an explosion on April 20 aboard the Deepwater Horizon rig which killed 11 people.
The oil stopped gushing from BP’s ruptured well in the Gulf of Mexico when it was capped on July 15 and engineers have since been working to permanently plug it. The damaged Macondo well has spewed about 4.9m barrels of oil into the gulf after an explosion on April 20 aboard the Deepwater Horizon rig which killed 11 people.
Entity-Driven Model

Documents

Entities

θ

Schema Distribution

Schema Type  (bombing)

Entity Role/Slot  (victim)

Entity Mentions  (observed)
Entity-Driven Model

Documents \( \theta \)

Entities \( t \)

Mentions \( s \)

Entity Mentions (observed)

\( \beta \), \( \delta \), \( P \), \( \kappa \), \( |T| \)

Schema Distribution

Schema Type (bombing)

Entity Role/Slot (victim)

Entity Mentions (observed)
Easy to Change Model Complexity
Learning and Inference

• Collapsed Gibbs Sampling
• Randomly initialize latent variables $s$ and $t$
• Sample $s$ and $t$ in sequence conditioned on full setting of other variables
• 1000 iterations
Gibbs Sampling

- Required to approximate the joint distribution.
- Most NLP graphical models are intractable, and require either sampling or other approximation techniques (i.e., simplifying the model)

- Major steps of a sampler:
  1. Initialize all latent variables with values based on data or intuition
  2. Compute conditional probabilities based on the values
  3. Relabel latent variables by sampling from the conditional distributions
  4. Repeat from (2)
Gibbs Sampling

• Practical steps of a sampler:
  1. Initialize all latent variables *to some value*
  2. Count everything (get your sufficient statistics)
  3. Relabel each latent variable one at a time based on your counts.
     a) Remove the variable’s value from your counts
     b) Compute that variable’s distribution based on all counts
     c) Sample the distribution from (b) and relabel the variable
     d) Update your counts with this new value.
Example

\[ S = \frac{\text{Count}(s=2, t=1)}{\text{Count}(s=*, t=1)} \]

\[ T = \begin{array}{ccccccccccc}
0 & 1 & 1 & 0 & 2 & 0 & 1 & 1 & 1 & 2 & 0 & 0 \\
1 & 1 & 2 & 3 & 1 & 0 & 1 & 2 & 2 & 3 & 3 & 0 \\
\end{array} \]
Example

\[
T = \begin{array}{cccccccccc}
0 & 1 & 1 & 0 & 2 & 0 & 1 & 1 & 1 & 2 & 0 & 0 \\
\end{array}
\]

\[
S = \begin{array}{cccccccccc}
1 & 1 & 2 & 3 & 1 & 0 & 1 & 2 & 2 & 3 & 3 & 0 \\
\end{array}
\]

\[
P(s=2 \mid t=1) = \frac{\text{Count}(s=2, t=1)}{\text{Count}(s=*, t=1)}
\]
Example

\[ S = \frac{\text{Count}(s=2, t=1)}{\text{Count}(s=*, t=1)} \]

\[ T = \frac{3}{5} \]

\[
\begin{align*}
P(s=2 \mid t=1) &= \text{Count}(s=2, t=1) / \text{Count}(s=*, t=1) \\
P(s=2 \mid t=1) &= 3 / 5
\end{align*}
\]
Example

\[ T = \begin{array}{cccccccccccccc}
0 & 1 & 1 & 0 & 2 & 0 & 1 & 1 & 1 & 2 & 0 & 0 \\
\end{array} \]

\[ S = \begin{array}{cccccccccccccc}
1 & 1 & 2 & 3 & 1 & 0 & 1 & 2 & 2 & 3 & 3 & 0 \\
\end{array} \]

\[
P(s=0 \mid t=1) = 0
\]

\[
P(s=1 \mid t=1) = \frac{2}{5}
\]

\[
P(s=2 \mid t=1) = \frac{3}{5}
\]

Sums to 1.0
Example

\[
T = \begin{array}{ccccccccc}
0 & 1 & 1 & 0 & 2 & 0 & 1 & 1 & 1 \\
1 & 1 & 2 & 3 & 1 & 0 & 1 & 2 & 2 \\
\end{array}
\]

\[
S = \begin{array}{ccccccccc}
1 & 1 & 2 & 3 & 1 & 0 & 1 & 2 & 3 \\
3 & 3 & 3 & 3 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
P(s=0 \mid t=1) = 0 \\
P(s=1 \mid t=1) = 3/5 \\
P(s=2 \mid t=1) = 2/5
\]

\[
P(s=0 \mid t=1) = 0.1 \\
P(s=1 \mid t=1) = 3.1/5.3 \\
P(s=2 \mid t=1) = 2.1/5.3
\]

Need smoothing!
Experiments

1. **Schema Quality**
   - Did we learn valid schemas/frames?

2. **Schema Extraction**
   - Do the learned schemas prove useful?
# Example Learned Schema

## Bombing Schema

### Victim
(Person 86%)

<table>
<thead>
<tr>
<th>Role</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>Object of kill</td>
</tr>
<tr>
<td>Guerrilla</td>
<td>Object of wound</td>
</tr>
<tr>
<td>Soldier</td>
<td>Subject of die</td>
</tr>
<tr>
<td>Man</td>
<td>Subject of blow up</td>
</tr>
<tr>
<td>Civilian</td>
<td>Subject of try</td>
</tr>
</tbody>
</table>

### Physical Target
(Object 65%)

<table>
<thead>
<tr>
<th>Object</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building</td>
<td>Object of destroy</td>
</tr>
<tr>
<td>Office</td>
<td>Object of damage</td>
</tr>
<tr>
<td>Explosive</td>
<td>Object of use</td>
</tr>
<tr>
<td>Station</td>
<td>Conjunct w/ office</td>
</tr>
<tr>
<td>Vehicle</td>
<td>Prep of number</td>
</tr>
</tbody>
</table>

### Instrument
(event 56%)

<table>
<thead>
<tr>
<th>Instrument</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bomb</td>
<td>Subject of explode</td>
</tr>
<tr>
<td>Explosion</td>
<td>Subject of occur</td>
</tr>
<tr>
<td>Attack</td>
<td>Object of cause</td>
</tr>
<tr>
<td>Charge</td>
<td>Object of place</td>
</tr>
<tr>
<td>Device</td>
<td>Subject of destroy</td>
</tr>
</tbody>
</table>
### Schema Quality

<table>
<thead>
<tr>
<th></th>
<th>Perp</th>
<th>Victim</th>
<th>Target</th>
<th>Instrument</th>
<th>Location</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Attack</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>2. Bombing</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>3. Kidnapping</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>4. Arson</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
</tr>
</tbody>
</table>

**Recall: 71%**
Schema Extraction

MUC-4 corpus, as before

Experiment Setup:
• Train on all 1700 documents
• Evaluate the inferred labels in the 200 test documents
• Repeat each experiment 30 times, average results
Evaluations

1. Flat Mapping
2. Schema Mapping

Mapping choice leads to very different extraction performance.
Evaluations

1. Flat Mapping
   - Map each learned slot to any MUC-4 slot

   ![Diagram showing mappings between schemas and events:]

   - **Schema 1**
     - Role 1
     - Role 2
     - Role 3
     - Role 4

   - **Schema 2**
     - Role 1
     - Role 2
     - Role 3
     - Role 4

   - **Schema 3**
     - Role 1
     - Role 2
     - Role 3
     - Role 4

   **Bombing**
   - Perpetrator
   - Victim
   - Target
   - Instrument

   **Arson**
   - Perpetrator
   - Victim
   - Target
   - Instrument
2. Schema Mapping

- Slots bound to a single MUC-4 template

- **Schema 1**
  - Role 1
  - Role 2
  - Role 3
  - Role 4

- **Schema 2**
  - Role 1
  - Role 2
  - Role 3
  - Role 4

- **Schema 3**
  - Role 1
  - Role 2
  - Role 3
  - Role 4

- **Bombing**
  - Perpetrator
  - Victim
  - Target
  - Instrument

- **Arson**
  - Perpetrator
  - Victim
  - Target
  - Instrument
## Results

### Schema Mapping Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Prec</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipelined (2011)</td>
<td>.48</td>
<td>.25</td>
<td>.33</td>
</tr>
<tr>
<td>Formal Schema Model</td>
<td>.42</td>
<td>.27</td>
<td>.33</td>
</tr>
</tbody>
</table>

*No improvement?*
## Results

### Schema Mapping Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Prec</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipelined (2011)</td>
<td>.48</td>
<td>.25</td>
<td>.33</td>
</tr>
<tr>
<td>Formal Schema Model</td>
<td>.42</td>
<td>.27</td>
<td>.33</td>
</tr>
</tbody>
</table>

---

No improvement?

### Tangible improvements

1. Far less training data. No external corpora.
2. Single stage training. Fewer parameters.
3. No separate extraction algorithm.
4. New feature integration is straight-forward.
Results

Flat Slot Mapping Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Prec</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cheung et al. (2013)</td>
<td>.32</td>
<td>.37</td>
<td>.34</td>
</tr>
<tr>
<td>Formal Schema Model</td>
<td>.41</td>
<td>.41</td>
<td>.41</td>
</tr>
</tbody>
</table>

event words

entity mention

entity role

schema type

event type

event words
# Results

## Flat Slot Mapping Evaluation

<table>
<thead>
<tr>
<th>Model</th>
<th>Prec</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cheung et al. (2013)</td>
<td>.32</td>
<td>.37</td>
<td>.34</td>
</tr>
<tr>
<td>Flat Relation Model</td>
<td>.26</td>
<td>.45</td>
<td>.33</td>
</tr>
<tr>
<td>Formal Schema Model</td>
<td>.41</td>
<td>.41</td>
<td>.41</td>
</tr>
</tbody>
</table>
## Results

### Flat Slot Mapping Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Prec</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cheung et al. (2013)</td>
<td>.32</td>
<td>.37</td>
<td>.34</td>
</tr>
<tr>
<td>Flat Relation Model</td>
<td>.26</td>
<td>.45</td>
<td>.33</td>
</tr>
<tr>
<td>Formal Schema Model</td>
<td>.41</td>
<td>.41</td>
<td>.41</td>
</tr>
</tbody>
</table>

### Gold Document Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Prec</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cheung et al. (2013)</td>
<td>.41</td>
<td>.44</td>
<td>.43</td>
</tr>
<tr>
<td>Formal Schema Model</td>
<td>.49</td>
<td>.43</td>
<td>.46</td>
</tr>
</tbody>
</table>
## Entity Role Performance

<table>
<thead>
<tr>
<th>Entity Role Performance</th>
<th>Prec</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perpetrator</td>
<td>.40</td>
<td>.20</td>
<td>.26</td>
</tr>
<tr>
<td>Victim</td>
<td>.42</td>
<td>.31</td>
<td>.34</td>
</tr>
<tr>
<td>Target</td>
<td>.38</td>
<td>.28</td>
<td>.31</td>
</tr>
<tr>
<td>Instrument</td>
<td>.57</td>
<td>.39</td>
<td>.45</td>
</tr>
</tbody>
</table>

Table 3: Results for each MUC-4 template slot using the template-mapping evaluation.
Other Models

Generative Proposals

• Cheung et al. (NAACL 2013)
• Bamman et al. (ACL 2013)
• Chambers (EMNLP 2013)
• Cheung and Penn (ACL 2013)
• Nguyen et al. (ACL 2015)
Latent Movie Characters

• Bamman, O’Connor, Smith (2013)
Latent Movie Characters
Neural Models?

(Modi and Titov, 2014)

• Idea: rather than learn a discrete set of events, learn a vector representation of the event

• Use a neural network model to learn this representation
  – Associate each argument and predicate with a vector embedding
  – These are connected in a feed-forward model to compute an overall event embedding
  – The units adjust their computation in order to minimize the error function
Events as word embeddings

• Neural network model:

• Used to rank the order in which events happen
  – Error function: how many pairs of events are incorrectly ordered?
Going Forward

• **Large-Scale Common-Sense Learning**
  – Massive knowledge mining
  – Quick
  – Not straightforward to use in reasoners?

• **Smaller-Scale Rich Generative Models**
  – Richer structure with a formal interpretation
  – Slow
  – Pre-cluster documents describing the same situation?
Conclusion

Minsky, on Frames (1974)

I try here to bring together several of these issues by pretending to have a unified, coherent theory.

These works raise more questions than they answer, and I have tried to note the theory's deficiencies.