Transformation-Based Learning

Advanced Statistical Methods in NLP
Ling 572
March 1, 2012
Roadmap

- Transformation-based (Error-driven) learning
  - Basic TBL framework
  - Design decisions

- TBL Part-of-Speech Tagging
  - Rule templates
  - Effectiveness

- HW #9
Transformation-Based Learning: Overview

- Developed by Eric Brill (~1992)
Transformation-Based Learning: Overview

- Developed by Eric Brill (~1992)
- Basic approach:
  - Apply initial labeling
  - Perform cascade of transformations to reduce error
Transformation-Based Learning: Overview

- Developed by Eric Brill (~1992)

- Basic approach:
  - Apply initial labeling
  - Perform cascade of transformations to reduce error

- Applications:
  - Classification
  - Sequence labeling
    - Part-of-speech tagging, dialogue act tagging, chunking, handwriting segmentation, referring expression generation
  - Structure extraction: parsing
TBL Architecture
Key Process

- Goal: Correct errors made by initial labeling
Key Process

- Goal: Correct errors made by initial labeling
- Approach: transformations
Key Process

- Goal: Correct errors made by initial labeling
- Approach: transformations
  - Two components:
Key Process

- **Goal**: Correct errors made by initial labeling
- **Approach**: transformations
  - Two components:
    - **Rewrite rule**:
      - e.g., MD $\rightarrow$ NN
Key Process

- **Goal**: Correct errors made by initial labeling
- **Approach**: transformations
  - Two components:
    - Rewrite rule:
      - e.g., MD $\rightarrow$ NN
    - Triggering environment:
      - e.g., prevT=DT
Key Process

- Goal: Correct errors made by initial labeling
- Approach: transformations
  - Two components:
    - Rewrite rule:
      - e.g., MD $\rightarrow$ NN
    - Triggering environment:
      - e.g., prevT=DT
  - Example transformation:
    - MD $\rightarrow$ NN if prevT=DT
    - The/DT can/NN rusted/VB
Key Process

- Goal: Correct errors made by initial labeling
- Approach: transformations
  - Two components:
    - Rewrite rule:
      - e.g., MD \(\rightarrow\) NN
    - Triggering environment:
      - e.g., prevT=DT
  - Example transformation:
    - MD \(\rightarrow\) NN if prevT=DT
    - The/DT can/MD rusted/VB \(\rightarrow\) The/DT can/NN rusted/VB
TBL: Training

1. Apply initial labeling procedure
TBL: Training

1. Apply initial labeling procedure

2. Consider all possible transformations, select transformation with best score on training data
TBL: Training

1. Apply initial labeling procedure

2. Consider all possible transformations, select transformation with best score on training data

3. Add rule to end of rule cascade
TBL: Training

1. Apply initial labeling procedure

2. Consider all possible transformations, select transformation with best score on training data

3. Add rule to end of rule cascade

4. Iterate over steps 2,3 until no more improvement
Error Reduction

Diagram showing the process of error reduction from an unannotated corpus to annotated corpus through multiple stages T1, T2, T3, T4, with decreasing error counts at each stage.
TBL: Testing

- For each test instance:
TBL: Testing

- For each test instance:
  1. Apply initial labeling
TBL: Testing

• For each test instance:

1. Apply initial labeling

2. Apply, in sequence, all matching transformations
Core Design Decisions
Core Design Decisions

- Choose initial-state annotator
  - Many possibilities:
Core Design Decisions

- Choose initial-state annotator
  - Many possibilities:
    - Simple heuristics, complex learned classifiers
Core Design Decisions

- Choose initial-state annotator
  - Many possibilities:
    - Simple heuristics, complex learned classifiers

- Specify legal transformations:
Core Design Decisions

- Choose initial-state annotator
  - Many possibilities:
    - Simple heuristics, complex learned classifiers

- Specify legal transformations:
  - ‘Transformation templates’: instantiate based on data
Core Design Decisions

- Choose initial-state annotator
  - Many possibilities:
    - Simple heuristics, complex learned classifiers

- Specify legal transformations:
  - ‘Transformation templates’: instantiate based on data
  - Can be simple and local or complex and long-distance
Core Design Decisions

- Choose initial-state annotator
  - Many possibilities:
    - Simple heuristics, complex learned classifiers

- Specify legal transformations:
  - ‘Transformation templates’: instantiate based on data
  - Can be simple and local or complex and long-distance

- Define evaluation function for selecting transform’n
Core Design Decisions

- Choose initial-state annotator
  - Many possibilities:
    - Simple heuristics, complex learned classifiers

- Specify legal transformations:
  - ‘Transformation templates’: instantiate based on data
  - Can be simple and local or complex and long-distance

- Define evaluation function for selecting transform’n
  - Evaluation measures:
Core Design Decisions

- Choose initial-state annotator
  - Many possibilities:
    - Simple heuristics, complex learned classifiers

- Specify legal transformations:
  - ‘Transformation templates’: instantiate based on data
  - Can be simple and local or complex and long-distance

- Define evaluation function for selecting transform’n
  - Evaluation measures: accuracy, f-measure, parse score
  - Stopping criterion:
Core Design Decisions

- Choose initial-state annotator
  - Many possibilities:
    - Simple heuristics, complex learned classifiers

- Specify legal transformations:
  - ‘Transformation templates’: instantiate based on data
  - Can be simple and local or complex and long-distance

- Define evaluation function for selecting transform’n
  - Evaluation measures: accuracy, f-measure, parse score
  - Stopping criterion: no improvement, threshold,..
More Design Decisions

- Issue:
  - Rule application can affect downstream application
More Design Decisions

- Issue:
  - Rule application can affect downstream application

- Approach:
  - Constraints on transformation application:
More Design Decisions

- **Issue:**
  - Rule application can affect downstream application

- **Approach:**
  - Constraints on transformation application:
    - Specify order of transformation application
More Design Decisions

- Issue:
  - Rule application can affect downstream application

- Approach:
  - Constraints on transformation application:
    - Specify order of transformation application
  - Decide if transformations applied immediately or after whole corpus is checked for triggering environments
Ordering Example

- Example sequence: A A A A A
Ordering Example

- Example sequence: A A A A A A
- Transformation: if prevT==A, A→B
Ordering Example

- Example sequence: A A A A A A
- Transformation: if prevT==A, A→B
- If check whole corpus for contexts before application:
Ordering Example

- Example sequence: A A A A A A
- Transformation: if prevT==A, A→B
- If check whole corpus for contexts before application:
  - A B B B B
- If apply immediately, left-to-right:
Ordering Example

- Example sequence: A A A A A A
- Transformation: if prevT==A, A→B
- If check whole corpus for contexts before application:
  - A B B B B
- If apply immediately, left-to-right:
  - A B A B A
- If apply immediately, right-to-left:
Ordering Example

- Example sequence: A A A A A
- Transformation: if prevT==A, A→B
- If check whole corpus for contexts before application:
  - A B B B B
- If apply immediately, left-to-right:
  - A B A B A
- If apply immediately, right-to-left:
  - A B B B B
Comparisons

- Decision trees:
  - Similar in
Comparisons

- Decision trees:
  - Similar in applying a cascade of tests for classification
  - Similar in interpretability

TBL:
Comparisons

- Decision trees:
  - Similar in applying a cascade of tests for classification
  - Similar in interpretability

- TBL:
  - More general: Can convert any DT to TBL
Comparisons

- Decision trees:
  - Similar in applying a cascade of tests for classification
  - Similar in interpretability

- TBL:
  - More general: Can convert any DT to TBL
  - Selection criterion
Comparisons

- Decision trees:
  - Similar in applying a cascade of tests for classification
  - Similar in interpretability

- TBL:
  - More general: Can convert any DT to TBL
  - Selection criterion: more focused: accuracy vs InfoGain
  - More powerful
Comparisons

- Decision trees:
  - Similar in applying a cascade of tests for classification
  - Similar in interpretability

- TBL:
  - More general: Can convert any DT to TBL
  - Selection criterion: more focused: accuracy vs InfoGain
  - More powerful: Can perform arbitrary output transform
    - Can refine arbitrary existing labeling
    - Multipass processing
  - More expensive:
Comparisons

- Decision trees:
  - Similar in applying a cascade of tests for classification
  - Similar in interpretability

- TBL:
  - More general: Can convert any DT to TBL
  - Selection criterion: more focused: accuracy vs InfoGain
  - More powerful: Can perform arbitrary output transform
    - Can refine arbitrary existing labeling
    - Multipass processing
  - More expensive:
    - Transformations tested/applied to all training data
Case Study: POS Tagging
TBL Part-of-Speech Tagging

- Design decisions:
TBL Part-of-Speech Tagging

- Design decisions:
  - Select initial labeling
  - Define transformation templates
  - Select evaluation criterion
  - Determine application order
POS Initial Labeling & Transformations

- Initial-state annotation?
POS Initial Labeling & Transformations

- Initial-state annotation?
- Heuristic:
POS Initial Labeling & Transformations

- Initial-state annotation?
  - Heuristic: Most likely tag for word in training corpus
POS Initial Labeling & Transformations

- Initial-state annotation?
  - Heuristic: Most likely tag for word in training corpus
    - What about unknown words? (In testing)
POS Initial Labeling & Transformations

- Initial-state annotation?
  - Heuristic: Most likely tag for word in training corpus
    - What about unknown words? (In testing)
      - Heuristic:
POS Initial Labeling & Transformations

- Initial-state annotation?
  - Heuristic: Most likely tag for word in training corpus
    - What about unknown words? (In testing)
      - Heuristic: Caps $\rightarrow$ NNP; lowercase $\rightarrow$ NN

- Transformation templates:
  - Rewrite rules:
POS Initial Labeling & Transformations

- Initial-state annotation?
  - Heuristic: Most likely tag for word in training corpus
    - What about unknown words? (In testing)
      - Heuristic: Caps $\rightarrow$ NNP; lowercase $\rightarrow$ NN

- Transformation templates:
  - Rewrite rules:
    - TagA $\rightarrow$ TagB
  - Trigger contexts:
POS Initial Labeling & Transformations

- Initial-state annotation?
  - Heuristic: Most likely tag for word in training corpus
    - What about unknown words? (In testing)
      - Heuristic: Caps $\rightarrow$ NNP; lowercase $\rightarrow$ NN

- Transformation templates:
  - Rewrite rules:
    - TagA $\rightarrow$ TagB
  - Trigger contexts: From 3 word windows
    - Non-lexicalized
    - Lexicalized
Non-lexicalized Templates

- No word references, only tags
- $t_{-1} == z$
Non-lexicalized Templates

- No word references, only tags
  - $t_{-1} = z$
  - $t_{+1} = z$
Non-lexicalized Templates

- No word references, only tags
  - $t_{-1} == z$
  - $t_{+1} == z$
  - $t_{-2} == z$
  - $t_{-1} == z$ or $t_{-2} == z$
  - $t_{+1} == z$ or $t_{+2} == z$ or $t_{+3} == z$
Non-lexicalized Templates

- No word references, only tags
  - \( t_{-1} == z \)
  - \( t_{+1} == z \)
  - \( t_2 == z \)
  - \( t_{-1} == z \) or \( t_2 == z \)
  - \( t_{+1} == z \) or \( t_{+2} == z \) or \( t_{+3} == z \)
  - \( t_{-1} == z \) and \( t_{+1} == w \)
  - \( t_{-1} == z \) and \( t_2 == w \)
  - ...

Lexicalized Templates

- Add word references as well as tags
- Use all non-lexicalized templates, plus
Lexicalized Templates

- Add word references as well as tags
- Use all non-lexicalized templates, plus
- $w_{-1} == w$
Lexicalized Templates

- Add word references as well as tags
  - Use all non-lexicalized templates, plus
    - $w_{-1} == w$
    - $w_{-2} == w$
    - $w_{-1} == w$ or $w_{-2} == w$
Lexicalized Templates

- Add word references as well as tags
- Use all non-lexicalized templates, plus
- \( w_1 = w \)
- \( w_2 = w \)
- \( w_1 = w \) or \( w_2 = w \)
- \( w_0 = w \) and \( w_1 = x \)
- \( w_0 = w \) and \( t_1 = z \)
- ....
Lexicalized Templates

- Add word references as well as tags
  - Use all non-lexicalized templates, plus
    - $w_{-1} == w$
    - $w_{-2} == w$
    - $w_{-1} == w \text{ or } w_{-2} == w$
    - $w_0 == w \text{ and } w_{-1} == x$
    - $w_0 == w \text{ and } t_{-1} == z$
    - $w_0 == w$
    - $w_{-1} == w \text{ and } t_{-1} == z$
    - ....
Remaining Design Decisions

- Optimization criterion:
Remaining Design Decisions

- Optimization criterion:
  - Greatest reduction in tagging error
  - Stops when no reduction beyond a threshold
Remaining Design Decisions

- Optimization criterion:
  - Greatest reduction in tagging error
  - Stops when no reduction beyond a threshold

- Ordering:
  - Processing is left-to-right

- Application:
Remaining Design Decisions

• Optimization criterion:
  • Greatest reduction in tagging error
  • Stops when no reduction beyond a threshold

• Ordering:
  • Processing is left-to-right

• Application:
  • Find all triggering environments first
  • Then apply transformations to all environments found
Top Non-Lexicalized Transformations

<table>
<thead>
<tr>
<th>#</th>
<th>Change Tag</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NN</td>
<td>Previous tag is TO</td>
</tr>
<tr>
<td>2</td>
<td>VBP</td>
<td>One of the previous three tags is MD</td>
</tr>
<tr>
<td>3</td>
<td>NN</td>
<td>One of the previous two tags is MD</td>
</tr>
<tr>
<td>4</td>
<td>VB</td>
<td>One of the previous two tags is DT</td>
</tr>
<tr>
<td>5</td>
<td>VBD</td>
<td>One of the previous three tags is VBZ</td>
</tr>
<tr>
<td>6</td>
<td>VBN</td>
<td>Previous tag is PRP</td>
</tr>
<tr>
<td>7</td>
<td>VBN</td>
<td>Previous tag is NNP</td>
</tr>
<tr>
<td>8</td>
<td>VBD</td>
<td>Previous tag is VBD</td>
</tr>
<tr>
<td>9</td>
<td>VBP</td>
<td>Previous tag is TO</td>
</tr>
<tr>
<td>10</td>
<td>POS</td>
<td>Previous tag is PRP</td>
</tr>
<tr>
<td>11</td>
<td>VB</td>
<td>Previous tag is NNS</td>
</tr>
<tr>
<td>12</td>
<td>VBD</td>
<td>One of previous three tags is VBP</td>
</tr>
</tbody>
</table>
Example Lexicalized Transformations

- if $w_{+2} == \text{as}, \text{IN} \rightarrow \text{RB}$
- PTB coding says: as tall as $\Rightarrow$ as/RB tall/JJ as/IN
- Initial tagging: as/IN tall/JJ as/IN
Example Lexicalized Transformations

• if $w_{+2} == \text{as, IN} \rightarrow \text{RB}$
  • PTB coding says: as tall as $\Rightarrow$ as/RB tall/JJ as/IN
  • Initial tagging: as/IN tall/JJ as/IN

• if $w_{-1} == \text{n’t}$ or $w_{-2} == \text{n’t}$, VBP $\rightarrow$ VB
  • Example: we do n’t eat or we do n’t usually sleep
TBL Accuracy

- Performance evaluated on different corpora:
  - Baseline: 92.4%
TBL Accuracy

- Performance evaluated on different corpora:
  - Penn WSJ: 96.6%
  - Penn Brown: 96.3%
  - Orig Brown: 96.5%
  - Baseline: 92.4%

- Lexicalized vs Non-lexicalized transformations
  - Non-lexicalized WSJ: 96.3%
  - Lexicalized WSJ: 96.6%
TBL Accuracy

- Performance evaluated on different corpora:
  - Penn WSJ: 96.6%
  - Penn Brown: 96.3%
  - Orig Brown: 96.5%
  - Baseline: 92.4%

- Lexicalized vs Non-lexicalized transformations
  - Non-lexicalized WSJ: 96.3%
  - Lexicalized WSJ: 96.6%

- Rules: 447 contextual rules (+ 243 for unknown wds)
TBL Summary

- Builds on an initial-state annotation
- Defines transformation templates for task
- Learns ordered list of transformations on labels
  - By iteratively selecting best transformation
- Classifies by applying initial-state and applicable transformations in order
Analysis

- Contrasts:
  - Distinctive learning strategy
Analysis

- Contrasts:
  - Distinctive learning strategy
    - Improves initial classification outputs
Analysis

- Contrasts:
  - Distinctive learning strategy
    - Improves initial classification outputs
  - Exploits current tagging state (previous, current, post) tags
Analysis

- **Contrasts:**
  - Distinctive learning strategy
    - Improves initial classification outputs
  - Exploits current tagging state (previous, current, post) tags
  - Can apply to classification, sequence learning, structure
  - Potentially global feature use – short vs long distance
    - Arbitrary transformations, rules
TBL: Advantages

- Pros:
TBL: Advantages

- Pros:
  - Optimization
TBL: Advantages

- Pros:
  - Optimization: Directly optimizes accuracy
TBL: Advantages

- Pros:
  - Optimization: Directly optimizes accuracy
  - Goes beyond classification to sequence & structure
    - Parsing, etc
  - Meta-classification:
TBL: Advantages

- **Pros:**
  - Optimization: Directly optimizes accuracy
  - Goes beyond classification to sequence & structure
    - Parsing, etc
  - Meta-classification:
    - Use initial state annotation
    - Explicitly manipulates full labeled output
      - Output of transformation is visible to other transformations
TBL: Disadvantages

- Cons:
TBL: Disadvantages

- Cons:
  - Computationally expensive:
    - Learns by iterating all possible template instantiations
    - Over all instances in the training data
      - Modifications have been proposed to accelerate
TBL: Disadvantages

- Cons

- Computationally expensive:
  - Learns by iterating all possible template instantiations
  - Over all instances in the training data
    - Modifications have been proposed to accelerate

- No probabilistic interpretation
  - Can’t produce ranked n-best lists
HW #9
TBL

- For Text Classification

- Transformations:
  - If (feature is present) and (classlabel == class1)
    - classlabel $\rightarrow$ class2
Q1

- Build a TBL trainer
  - Input: training_data file
  - Output: model_file
    - Lines:
      - featureName from_class to_class gain
    - Parameter: Gain threshold

- Example:
  - guns
  - talk guns mideast 89
Q2

- TBL decoder
  - Input: test_file model_file
  - Output: output_file
  - Parameter: N → # of transformations

- Output_file:
  - InstanceID goldLabel outputLabel rule1 rule2 ...
  - Rule: feature from_class to_class

- Ex:
  - inst1 guns mideast we guns misc talk misc mideast