Maximum Metric Score Training for Coreference Resolution

Shanheng Zhao and Hwee Tou Ng
National University of Singapore
23 August 2010
Beijing
Outline

• Introduction
• Motivation
• Contribution
• Related Work
• Evaluation Metrics
• Maximum Metric Score Training
• Experiments
• Conclusion
Introduction

- **Coreference Resolution**
  - Task definition: The process of determining whether two or more noun phrases (NPs) in a text refer to the same entity
  - A typical coreference resolution system:
    - *Two-class classification*
    - *Machine learning-based*
    - *Training: minimize the number of misclassified training instances*
    - *Testing: maximize either the local or the global probability of the coreferential relation assignments*
Motivation

- Minimizing the number of misclassified training instances during training does **NOT** guarantee maximizing the F-measure of the chosen evaluation metric for coreference resolution
  - Coreference is a rare relation
  - Evaluation metrics for coreference resolution are based on global assignments
  - The extracted training instances are not equally easy to be classified, and are not equally important
Contribution

- Propose maximum metric score training (MMST), a novel approach
  - Perform maximization of the chosen evaluation metric score on the training corpus during training
  - Iteratively assign higher weights to the hard-to-classify training instances
  - Output a standard classifier
  - Use of instance weighting and beam search
  - Significantly improve over the baselines
  - Report improved results over the state-of-the-art on five standard benchmark corpora and two evaluation metrics
Related Work

• Framework
  • Soon et al. (2001) and Ng & Cardie (2002b) proposed the most widely used training and testing framework for coreference resolution; our baseline system follows a similar framework

• Maximize metric score during testing
  • E.g. Ng (2005) -- time-consuming for large data sets

• Maximize metric score during training (neg.)
  • Wick & McCallum (2009) found that training on classification accuracy, in most cases, outperformed training on the coreference evaluation metric
Related Work (cont.)

• **Maximizing metric score during training**
  • Rule pruning (Ng & Cardie (2002a))
  • Varying different components to maximize metric score on a held-out development set (Ng (2004))
  • Approximate optimizing the ACE metric (Daume III (2006))
  • Document-level boosting (Vemulapalli et al (2009))
Evaluation Metrics

• Definitions
  • *Key*: the gold standard annotation
  • *Response*: the output by a coreference resolution system
  • *Mention/Markable*: an NP which satisfies the markable definition
  • *Link*: a pair of coreferential markables
  • *Singleton*: a markable which has no links to other markables
  • *Coreference chain*: a set of coreferential markables
Evaluation Metrics (cont.)

- **MUC Evaluation Metric (Vilain et al (1995))**

\[
Recall = \frac{\sum(|S_i| - |p(S_i)|)}{\sum(|S_i| - 1)}
\]

- \(S_i\): a coreference chain generated by the key
- \(p(S_i)\): a partition of \(S_i\) relative to the response
- Precision is defined similarly
- F-measure is a trade-off between recall and precision

\[
F = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision}
\]
Evaluation Metrics (cont.)

• B-CUBED Evaluation Metric (Bagga & Baldwin (1998))

\[
Recall = \frac{1}{N} \sum_{d \in D} \sum_{m \in d} \frac{|O_m|}{|S_m|}
\]

- \( D \): set of documents
- \( d \): a document
- \( m \): a markable
- \( S_m \): coreference chain in the key that contains \( m \)
- \( O_m \): overlap of \( S_m \) and the coreference chain in the response that contains \( m \)
- \( N \): total number of markables in \( D \)
- Precision is defined similarly
- F-measure is a trade-off between recall and precision

\[
F = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision}
\]
Evaluation Metrics (cont.)

**key**

- - - -

**response**

- - - - -

**MUC**

\[
Recall = \frac{4 - 2}{4 - 1} = 66.7\% \\
Precision = \frac{(1 - 1) + (3 - 1)}{(1 - 1) + (3 - 1)} = 100\% \\
F = 80\%
\]

**B-CUBED**

\[
Recall = \frac{1}{4} \left( \frac{1}{4} + \frac{3}{4} + \frac{3}{4} + \frac{3}{4} \right) = 62.5\% \\
Precision = \frac{1}{4} \left( \frac{1}{3} + \frac{3}{3} + \frac{3}{3} + \frac{3}{3} \right) = 100\% \\
F = 76.9\%
\]
Maximum Metric Score Training

• Coreference Clustering
  • Same as in most prior work
  • Pair-wise classifier does not guarantee transitivity
  • All predicted (positive) coreferential pairs are clustered together

• Note: not all positive pairs will be predicted by the classifier. It depends on the resolution strategy. For example, the closest-first method will stop after the closest antecedent is found (Soon et al (2001))
MMST (cont.)

- Suppose there are $m_k$ and $m_r$ coreferential links in the key and response, respectively, and a coreference resolution system successfully predicts $n$ correct links.

\[
\text{Recall} = \frac{n}{m_k} \quad \quad \text{Precision} = \frac{n}{m_r}
\]

- Wrongly classified instances
  - False positive
  - False negative
MMST (cont.)

- False Positive

\[
\text{Recall} = \frac{n}{m_κ} \quad \Rightarrow \quad \text{Recall} = \frac{n}{m_κ}
\]

\[
\text{Precision} = \frac{n}{m_r} \quad \Rightarrow \quad \text{Precision} = \frac{n}{m_r - 1}
\]
MMST (cont.)

• False Negative
  • If the two markables are in the same coreference chain after clustering

\[
\text{Recall} = \frac{n}{m_k} \quad \rightarrow \quad \text{Recall} = \frac{n}{m_k}
\]

\[
\text{Precision} = \frac{n}{m_r} \quad \rightarrow \quad \text{Precision} = \frac{n}{m_r}
\]
MMST (cont.)

• False Negative
  • If the two markables are NOT in the same coreference chain after clustering

\[
\text{Recall} = \frac{n}{m_k} \quad \text{versus} \quad \text{Recall} = \frac{n + 1}{m_k}
\]

\[
\text{Precision} = \frac{n}{m_r} \quad \text{versus} \quad \text{Precision} = \frac{n + 1}{m_r + 1}
\]
MMST (cont.)

• Instruct the learning algorithm to pay more attention to the false positive and false negative instances, and predict them correctly by assigning them *more weight*
  • Use all possible pairs of training instances
  • Initially, all pairs are equally weighted
  • Iteratively assign more weights to the hard-to-classify pairs using a beam search algorithm
MMST (cont.)

• **Beam Search**
  
  • Goal: a set of weights to assign to the training instances s.t. the classifier trained on the weighted training instances gives the maximum coreference metric score when evaluated on the training instances

• **Search state**
  
  • A set of weighted training instances
  • A classifier trained on the weighted training instances
  • F-measure of the classifier when evaluated on the training instances using the chosen coreference evaluation metric

• Root (initial state): all training instances are equally weighted
MMST (cont.)

• Binary Search Tree
  • Each search state is a node in the binary search tree
  • Adding weights to false positive training instances
  • Adding weights to false negative training instances
MMST (cont.)

• **Beam Search**
  - All nodes are sorted in descending order of F-measure
  - Only top $M$ nodes are kept
  - Discarded nodes can be *leaf* nodes or *non-leaf* nodes

• **Stopping Criteria**
  - When all the nodes in the beam are *non-leaf* nodes
    - All the nodes in the beam have been expanded
    - Expanding the nodes in the beam will not improve the evaluation metric score
The Algorithm

1. Initialization

2. For each unexpanded beam node
   1) Classify (Predict)
   2) Update false positive
   3) Update false negative
   4) Beam pruning

3. Repeat Step 2 until all nodes in the beam have been expanded

4. Return best classifier
The Algorithm (cont.)

1. Initialization
   i. Assign an equal weight of 1 to all training instances ($w_{ij}=1$)
   ii. Train a classifier
   iii. Evaluate it and get the corresponding F-measure
   iv. Put it into the beam

2. For each unexpanded beam node
   1) Classify (Predict)
   2) Update false positive
   3) Update false negative
   4) Beam pruning

3. Repeat Step 2 until all nodes in the beam have been expanded

4. Return best classifier
The Algorithm (cont.)

1. Initialization

2. For each unexpanded beam node
   1) Classify (Predict)
      i. Use the classifier associated with the node to classify all training instances
   2) Update false positive
   3) Update false negative
   4) Beam pruning

3. Repeat Step 2 until all nodes in the beam have been expanded

4. Return best classifier
The Algorithm (cont.)

1. **Initialization**

2. **For each unexpanded beam node**
   1) Classify (Predict)
   2) Update false positive
      i. For all false positive instances: \( w'_{ij} = w_{ij} + \delta \)
      ii. Train a new classifier with the weighted instances
      iii. Evaluate it and get the corresponding F-measure
      iv. Put it into the beam
   3) Update false negative
   4) Beam pruning

3. **Repeat Step 2 until all nodes in the beam have been expanded**

4. **Return best classifier**
The Algorithm (cont.)

1. Initialization

2. For each unexpanded beam node
   1) Classify (Predict)
   2) Update false positive
   3) Update false negative
      i. For all false negative instances in which the two markables are NOT in the same coreference chain: $w'_{ij} = w_{ij} + \delta$
      ii. Train a new classifier with the weighted instances
      iii. Evaluate it and get the corresponding F-measure
      iv. Put it into the beam
   4) Beam pruning

3. Repeat Step 2 until all nodes in the beam have been expanded

4. Return best classifier
The Algorithm (cont.)

1. **Initialization**

2. **For each unexpanded beam node**
   1) Classify (Predict)
   2) Update false positive
   3) Update false negative
   4) Beam pruning
      i. Sort the beam nodes in descending order of F-measure
      ii. Only top $M$ nodes are kept

3. **Repeat Step 2 until all nodes in the beam have been expanded**

4. **Return best classifier**
The Algorithm (cont.)

1. **Initialization**

2. **For each unexpanded beam node**
   1) Classify (Predict)
   2) Update false positive
   3) Update false negative
   4) Beam pruning

3. **Repeat Step 2 until all nodes in the beam have been expanded**

4. **Return best classifier**
The Algorithm (cont.)

1. Initialization

2. For each unexpanded beam node
   1) Classify (Predict)
   2) Update false positive
   3) Update false negative
   4) Beam pruning

3. Repeat Step 2 until all nodes in the beam have been expanded

4. Return best classifier
Experiments

• Experimental Setup
  • The corpora
    • Two MUC data sets: MUC6 and MUC7
    • Three ACE data sets: BNEWS, NPAPER, and NWIRE

<table>
<thead>
<tr>
<th># Docs</th>
<th>MUC6</th>
<th>MUC7</th>
<th>BNEWS</th>
<th>NPAPER</th>
<th>NWIRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>30</td>
<td>30</td>
<td>216</td>
<td>76</td>
<td>130</td>
</tr>
<tr>
<td>Test</td>
<td>30</td>
<td>20</td>
<td>51</td>
<td>17</td>
<td>29</td>
</tr>
</tbody>
</table>

• Raw text input
• Implemented on top of BART (Versley et al (2008))
Experiments (cont.)

• Experimental Setup
  • Evaluation metric
    • MUC
    • B-CUBED
      • $B^3all$: retains all twinless markables (Stoyanov et al (2009))
Experiments (cont.)

• **Baseline System**
  • In the literature
    • Versley et al (2008)
    • Ng (2004)
    • Ng (2005)
  • Our implemented baselines
    • Identical feature set as Versley et al (2008), except the tree-valued and string-valued features
    • J48, the WEKA implementation of C4.5 decision tree
    • SNL-style baseline (Soon et al. (2001))
    • All-style baseline
Experiments (cont.)

• Parameter Tuning
  • Parameters: $M$ & $\delta$
  • Development training: 2/3 of the training corpus
  • Development test: 1/3 of the training corpus
  • Fixed $\delta = 1$, evaluate $M = 2, 4, 6, \ldots, 20$
  • Fixed $M$, evaluate $\delta = 0.1, 0.2, 0.3, \ldots, 2.0$
Experiments (cont.)

- Parameter Tuning on MUC6

- Best: $M=4$ or $M=6$

- $\delta=1.0$
Experiments (cont.)

- Results with MUC evaluation metric

<table>
<thead>
<tr>
<th>Model</th>
<th>MUC6</th>
<th>MUC7</th>
<th>BNEWS</th>
<th>NPAPER</th>
<th>NWIRE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>P</td>
<td>F</td>
<td>R</td>
<td>P</td>
</tr>
<tr>
<td>Versley et al 08</td>
<td>60.7</td>
<td>65.4</td>
<td>63.0</td>
<td>64.1</td>
<td>67.7</td>
</tr>
<tr>
<td>Ng 04</td>
<td>75.8</td>
<td>61.4</td>
<td>67.9</td>
<td>64.2</td>
<td>60.2</td>
</tr>
<tr>
<td>SNL Baseline</td>
<td>67.0</td>
<td>49.2</td>
<td>56.7</td>
<td>63.0</td>
<td>54.2</td>
</tr>
<tr>
<td>All Baseline</td>
<td>56.9</td>
<td>69.2</td>
<td>62.5</td>
<td>51.5</td>
<td>73.4</td>
</tr>
<tr>
<td>MMST</td>
<td>73.3</td>
<td>59.9</td>
<td>65.9</td>
<td>66.8</td>
<td>59.8</td>
</tr>
</tbody>
</table>

\[ M=6, \delta=1.0 \] \[ M=6, \delta=0.7 \] \[ M=6, \delta=1.8 \] \[ M=6, \delta=0.9 \] \[ M=14, \delta=0.7 \]
Experiments (cont.)

- Results with B-CUBED evaluation metric

<table>
<thead>
<tr>
<th>Model</th>
<th>MUC6 R</th>
<th>MUC6 P</th>
<th>MUC6 F</th>
<th>MUC7 R</th>
<th>MUC7 P</th>
<th>MUC7 F</th>
<th>BNEWS R</th>
<th>BNEWS P</th>
<th>BNEWS F</th>
<th>NPAPER R</th>
<th>NPAPER P</th>
<th>NPAPER F</th>
<th>NWIRE R</th>
<th>NWIRE P</th>
<th>NWIRE F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ng 05</td>
<td></td>
<td></td>
<td></td>
<td>57.0</td>
<td>77.1</td>
<td>65.6</td>
<td>62.8</td>
<td>71.2</td>
<td>66.7</td>
<td>59.3</td>
<td>75.4</td>
<td>66.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SNL Baseline</td>
<td>57.8</td>
<td>74.4</td>
<td>65.1</td>
<td>57.6</td>
<td>76.5</td>
<td>65.7</td>
<td>62.0</td>
<td>74.7</td>
<td>67.8</td>
<td>61.8</td>
<td>70.4</td>
<td>65.8</td>
<td>65.8</td>
<td>75.9</td>
<td>70.5</td>
</tr>
<tr>
<td>All Baseline</td>
<td>51.6</td>
<td>86.3</td>
<td>64.6</td>
<td>49.1</td>
<td>90.1</td>
<td>63.6</td>
<td>61.6</td>
<td>83.7</td>
<td><strong>71.0</strong></td>
<td>63.9</td>
<td>74.0</td>
<td>68.6</td>
<td>64.8</td>
<td>80.1</td>
<td><strong>71.7</strong></td>
</tr>
<tr>
<td>MMST</td>
<td>62.7</td>
<td>81.5</td>
<td><strong>70.9</strong></td>
<td>61.8</td>
<td>73.6</td>
<td><strong>67.2</strong></td>
<td>61.6</td>
<td>83.7</td>
<td><strong>71.0</strong></td>
<td>63.1</td>
<td>76.2</td>
<td><strong>69.1</strong></td>
<td>64.3</td>
<td>81.0</td>
<td><strong>71.7</strong></td>
</tr>
</tbody>
</table>

$M=6$, $\delta=1.0$

$M=8$, $\delta=0.8$

$M=6$, $\delta=0.9$

$M=14$, $\delta=0.5$

$M=6$, $\delta=0.1$
Experiments (cont.)

- It takes 6-9 iterations for MMST to stop.
- The number of explored states in the binary search tree for all the test data sets, including the root, is 33, 39, 25, 29, and 75 on MUC6, MUC7, BNEWS, NPAPER, NWIRE, respectively.
- On the MUC6 data set, it takes 1.6 hours and 31 seconds for training and testing, respectively, on an Intel Xeon 2.33GHz machine.
Conclusion

• We present a novel maximum metric score training approach comprising the use of instance weighting and beam search
• Experimental results show that the approach achieves significant improvements over the baseline systems on most test data sets
• The proposed approach improves upon state-of-the-art results on most of the five standard benchmark corpora with both the MUC and the B-CUBED evaluation metrics
THANK YOU