BCMI-NLP Labeled-Alignment-Based Entailment System for NTCIR-10 RITE-2 Task

Xiao-Lin Wang, Hai Zhao, and Bao-Liang Lu
Center for Brain-like Computing and Machine Intelligence
MOE-Microsoft Key Laboratory for Intelligent Computing and Intelligence Systems
Department of Computer Science and Engineering,
Shanghai Jiao Tong University
arthur.xl.wang@gmail.com {zhaohai,blu}@cs.sjtu.edu.cn
Overview

• Alignment-based RITE approach

• Labeled alignment scheme

• Labeled-alignment-based RITE approach

• Formal runs
Alignment-based RITE approach

- Intuition: sufficiently good alignment → close lexical and structural similarity → entailment relation

Read into → interpreted
what he wanted → in his own way
NULL → just

Therefore, entailment
Alignment-based RITE approach

• Shortage:
  – originally developed for machine translation
  – not justify non-entailment pairs 😞

Critical non-aligned:
  flood → NULL
  NULL → ferry sinking

Non-critical non-aligned:
  in → NULL
  NULL → caused that

mixed, so hard to predict
Labeled alignment scheme

- Augment the normal alignment with negative links

Roadmap:
1) manually annotate
2) train an alignment model
3) train an RITE predict model

Negative links:
- flood $\nrightarrow$ ferry sinking

Therefore, non-entailment
Labeled alignment scheme

• More examples

Mismatched numbers

Different named entities
Labeled alignment scheme

• More examples
Labeled-alignment-based RITE approach

normal-alignment-based approach

labeled-alignment-based approach
### Labeled-alignment-based RITE

#### Link Type Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whether (e_1) and (e_2) are in an antonym list</td>
<td></td>
</tr>
<tr>
<td>Whether (e_1) and (e_2) are in an synonym list</td>
<td></td>
</tr>
<tr>
<td>Whether (e_1) and (e_2) are unequal numbers</td>
<td></td>
</tr>
<tr>
<td>Whether (e_1) and (e_2) are different named entities</td>
<td></td>
</tr>
<tr>
<td>Relation of (e_1) and (e_2) in an ontology (hyponym, sibling, etc.)</td>
<td></td>
</tr>
<tr>
<td>Ontology-based similarities of (e_1) and (e_2)</td>
<td></td>
</tr>
<tr>
<td>Count of common characters</td>
<td></td>
</tr>
<tr>
<td>Length of the common prefixes</td>
<td></td>
</tr>
<tr>
<td>Length of the common suffix</td>
<td></td>
</tr>
<tr>
<td>Tuple of the syntactic tags</td>
<td></td>
</tr>
<tr>
<td>Tuple of the ancestors in an ontology</td>
<td></td>
</tr>
<tr>
<td>Tuple of whether (e_1) or (e_2) is in a list of negative expressions</td>
<td></td>
</tr>
<tr>
<td>Tuple of whether (e_1) or (e_2) is the head of a noun phrase</td>
<td></td>
</tr>
</tbody>
</table>
Labeled-alignment-based RITE

• Classification component
  ◦ Sample representation: single flat vector that combines the features extracted from all the links
  ◦ Classifier: RBF-kerneled SVM (1-vs-rest for MC)

• Open question: why a cascaded system that
  - first classifies the link type
  - then classifies the RITE relation
  performs poorly? (worse than baseline)
Labeled–alignment-based RITE

• Possible answer for the failure of a cascaded system: not robust enough under current (public) available resources

For example:

三峡旅游不再有淡旺季的问题
三峡旅游不再有枯水期的问题

– Hard to accurately decide whether two normal Chinese words are antonym or not with the knowledge base of CiLin and mini-Hownet.
Formal Run

• System Implement
  – automated supervised-learning aligner
    (N. chambers et al., 2007; B. MacCartny et al., 2008)

  – Manually annotate the training data set
    • alignment links
    • positive/negative labels
Formal Run -- aligner

**Inputs**
- training problems \( \langle P_j, H_j \rangle, j = 1..n \)
- corresponding gold-standard alignments \( E_j \)
- a number of learning epochs \( N \) (e.g. 50)
- a “burn-in” period \( N_0 < N \) (e.g. 10)
- initial learning rate \( R_0 \) (e.g. 1) and multiplier \( r \) (e.g. 0.8)
- a vector of feature functions \( \Phi(E) \)
- an alignment algorithm \( \text{ALIGN}(P, H; w) \) which finds a good alignment for \( \langle P, H \rangle \) using weight vector \( w \)

**Initialize**
- Set \( w = 0 \)

**Repeat** for \( i = 1 \) to \( N \)
- Set \( R_i = r \cdot R_{i-1} \), reducing the learning rate
- Randomly shuffle the training problems
- For \( j = 1 \) to \( n \):
  - Set \( \hat{E}_j = \text{ALIGN}(P_j, H_j; w) \)
  - Set \( w = w + R_i \cdot (\Phi(E_j) - \Phi(\hat{E}_j)) \)
- Set \( w = w / \|w\|_2 \) (L2 normalization)
- Set \( w[i] = w \), storing the weight vector for this epoch

**Return** an averaged weight vector:
- \( w_{avg} = 1 / (N - N_0) \sum_{i=N_0+1}^{N} w[i] \)
Formal Run – aligner (decoding)

Inputs
- an alignment problem \((P, H)\)
- a number of iterations \(N\) (e.g. 100)
- initial temperature \(T_0\) (e.g. 40) and multiplier \(r\) (e.g. 0.9)
- a bound on edit size \(max\) (e.g. 6)
- an alignment scoring function, \(\text{SCORE}(E)\)

Initialize
- Let \(E\) be an “empty” alignment for \((P, H)\) (containing only DEL and INS edits, no EQ or SUB edits)
- Set \(\hat{E} = E\)

Repeat for \(i = 1\) to \(N\)
- Let \(\{F_1, F_2, \ldots\}\) be the set of possible successors of \(E\).
  To generate this set:
  - Consider every possible edit \(f\) up to size \(max\)
  - Let \(C(E, f)\) be the set of edits in \(E\) which “conflict” with \(f\) (i.e., involve at least some of the same tokens as \(f\))
  - Let \(F = E \cup \{f\} \setminus C(E, f)\)
- Let \(s(F)\) be a map from successors of \(E\) to scores generated by \(\text{SCORE}\)
- Set \(p(F) = \exp s(F)\), and then normalize \(p(F)\), transforming the score map to a probability distribution
- Set \(T_i = r \cdot T_{i-1}\)
- Set \(p(F) = p(F)^{1/T_i}\), smoothing or sharpening \(p(F)\)
- Renormalize \(p(F)\)
- Choose a new value for \(E\) by sampling from \(p(F)\)
  If \(\text{SCORE}(E) > \text{SCORE}(\hat{E})\), set \(\hat{E} = E\)

Return \(\hat{E}\)

Stimulated annealing algorithm

“Possible edit” means “possible alignment link”

Generated from:
1) Each segmented word in \(t_1\) \(\rightarrow\) Each segmented word in \(t_2\)
2) Each syntactic node in \(t_1\) \(\rightarrow\) Each syntactic node in \(t_2\)
3) Each NE in \(t_1\) \(\rightarrow\) Each NE in \(t_2\)
4) Expression \(e_1\) in \(t_1\) \(\rightarrow\) Expression \(e_2\) in \(t_2\) where,
   \((e_1, e_2)\) appears in a synonym list,
   or a antonym list,
   or in Cilin,
   or in hownet,
   or other resources
Formal Run

• System Implement
  – Segment: BaseSeg (Zhao and Kit, 2006)
  
  – Syntactic parser: Stanford factored parser
  
  – NER: BaseNER (zhao and Kit, 2008)
  
  – Chinese ontologies: Cilin (Mei et al.,1983; Luo, 2007), Hownet (Dong, 2003; Liu and Li, 2002; Liu and Singh, 2004)
  
  – Classifier: LibSVM (Lin, 2011)
Formal Run

<table>
<thead>
<tr>
<th>Run</th>
<th>Method</th>
<th>Macro-F1 on BC</th>
<th>Macro-F1 on MC</th>
<th>Worse. Rank. on RITE4QA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run01</td>
<td>Char-overlap</td>
<td>67.04</td>
<td>39.95</td>
<td>2.67*</td>
</tr>
<tr>
<td>Run02</td>
<td>Normal-align.</td>
<td>66.89</td>
<td>44.88</td>
<td>0.00*</td>
</tr>
<tr>
<td>Run03</td>
<td>Labeled-align.</td>
<td>73.84</td>
<td>56.82</td>
<td>8.00*</td>
</tr>
</tbody>
</table>

* not checked during formal run due to limit of time
References


References


Thank you