Dependency-directed Tree Kernel-based Protein-Protein Interaction Extraction from Biomedical Literature

Longhua Qian and Guodong Zhou
NLP Lab
Soochow University, Suzhou, China
9 Nov. 2011
IJCNLP 2011, ChiangMai, Thailand
Outline

1. Introduction
2. Related work
3. Constituent structure representation
4. Experimentation
5. Conclusion and future work
1. Introduction

**background**
- determining protein interaction partners is crucial to understand both the functional role of individual proteins and the organization of the entire biological process

**Approach**
- Manual collection of relevant PPI information from thousands of biomedical research papers is prohibitive that automatic extraction approaches with the help of NLP techniques become necessary.
**Task definition**

- PPI extraction
  Determining whether a relationship exists between a pair of protein names occurring in a sentence from the biomedical perspective.
- For example
  The ability of **PROT1** to interact with the **PROT2** was detected.

Denotes an interaction between **PROT1** and **PROT2**
2. Related work (kernel-based)

- **PPI extraction**
  - a generalized substring kernel over a mixture of words and word classes (Bunescu et al., 2005)
  - cosine similarity and edit distance among dependency paths (Erkan et al., 2007)
  - a tree kernel over dependency structures (Sætre et al., 2007)
  - all-dependency-paths graph kernel (Airola et al., 2008)
  - a walk-weighted subsequence kernel based on shortest dependency paths (Kim et al., 2010)

- **Semantic relation extraction in news domain**
  - kernels-based methods (Zelenko et al., 2003; Culotta and Sorensen, 2004; Bunescu et al., 2005b)
  - convolution tree kernels over constituent parse trees (Zhang et al., 2006; Zhou et al., 2007a; Qian et al., 2008)

- **However**
  - Phrase structure tree (Miyao et al., 2008; Tikk et al., 2010) performs worse than other representations in PPI extraction.
3. Constituent structure representation

1. Currently used constituent structures

2. Shortest Dependency Path- Constituent Path Tree
Currently used constituent structures

- 5 constituent parse trees
- **MCT** (Minimum Complete Tree) (Zhang et al., 2006)
- **SPT** (Shortest Path-enclosed Tree) (Zhang et al., 2006)
- **CS-SPT** (Context-Sensitive Shortest Path-enclosed Tree) (Zhou et al., 2007)
- **DSPT** (Dynamic Syntactic Parse Tree) (Qian et al., 2008)
Fig 1. Various constituent structures for the sentence “Association between \textit{PROT1} and cyclin B1 / \textit{PROT2} was detected in the HeLa cells.”
Disadvantages

- MCT contains much useful information as well as much noise;
- SPT delete some noise outside the shortest path while retaining some noise inside this path;
- CS-SPT extends a part of useful predicate-related information;
- DSPT dynamically prune noise according to heuristics, but at the cost of adaptability and granularity.
Out SDP-CPT solution

- Considering the importance of dependency path in PPI extraction
- the effectiveness of employing dependency information for tree kernel-based semantic relation extraction in the newswire domain
- Shortest Dependency Path-directed Constituent Parse Tree
  reshape the constituent parse tree by making use of the shortest dependency path between two proteins.
**SDP-CPT generation algorithm**

**Input:** a sentence and two proteins in it  
**Output:** an SDP-CPT  

**Steps:**
1) **Parsing:** Given the input sentence, generate the constituent parse tree using a constituent parser, and various dependency tuples using a dependency parser.
2) **CDP, SDP extraction:** Given the two proteins, extract the shortest constituent path (SCP, i.e. the shortest path-enclosed tree) from the constituent parse tree and construct the shortest dependency path (SDP) from the dependency tuples.
3) **Addition:** For each word along the SDP, add the corresponding leaf word node and its upper constituents to the SCP. Particularly,
   - when the dependency type is “prep_xx”, such as “prep_of”, the preposition xx and its associated constituents are also added;
   - when the word to be added is outside the SCP, a new path from the current lowest common ancestor to one of the added words’ ancestors is also added.
4) **Merging:** Merge any two consecutive NP/VP nodes along the paths into a single one.
An example of SDP-CPT

(a) the Shortest Dependency Path (SDP)

(b) the Shortest Constituent Path (SCP)

(c) An Example of SDP-CPT
4. Experimentation

- Corpora and preprocessing

5 PPI corpora:

- AIMed (Bunescu et al., 2005a), BioInfer (Pyysalo et al., 2007), HPRD50 (Fundel et al., 2007), IEPA (Ding et al., 2002) and LLL (Nédellec, 2005).

Preprocessing

1. pair-wise PPI candidate instances are generated in a sentence two protein names are blinded by PROT1 and PROT2 respectively
2. sentences are parsed using the Stanford Parser to generate both the constituent parse trees and dependency tuples.
Classifier and Evaluation Metrics

- Classifier tools
  SVMlight–TK (Moschitti, 2004)
- Evaluation
  OAOD (One Answer per Occurrence in the Document) with 10-fold document-level cross-validation
- Evaluation Metrics
  Precision (P), Recall (R) and harmonic F1-score (F1) as well as AUC score
Experimental Results

- 1. Comparison of different lengths of dependency paths on the AIMed corpus
- 2. Contribution of different kinds of dependencies on the AIMed corpus
- 3. Comparison of different constituent parse tree structures across major PPI corpora
- 4. Comparison of kernel-based PPI extraction systems on the AIMed corpus
Experimental Results 1

Comparison of different lengths of dependency paths on the AIMed corpus

<table>
<thead>
<tr>
<th>Length</th>
<th>P (%)</th>
<th>R (%)</th>
<th>F1</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPT</td>
<td>57.0</td>
<td>40.7</td>
<td>47.1</td>
<td>79.9</td>
</tr>
<tr>
<td>SCP+L0 (SCP)</td>
<td>45.0</td>
<td>19.5</td>
<td>26.5</td>
<td>67.9</td>
</tr>
<tr>
<td>SCP+L1</td>
<td>59.7</td>
<td>45.8</td>
<td>51.4</td>
<td>80.2</td>
</tr>
<tr>
<td>SCP+L2</td>
<td>59.2</td>
<td>51.7</td>
<td>55.0</td>
<td>82.3</td>
</tr>
<tr>
<td>SCP+L3</td>
<td>58.0</td>
<td>51.9</td>
<td>54.6</td>
<td>82.2</td>
</tr>
<tr>
<td>SCP+L4</td>
<td>59.3</td>
<td>54.0</td>
<td>56.2</td>
<td>82.6</td>
</tr>
<tr>
<td>SDP-CPT</td>
<td><strong>59.6</strong></td>
<td><strong>54.3</strong></td>
<td><strong>56.7</strong></td>
<td><strong>82.7</strong></td>
</tr>
</tbody>
</table>

*The words with different length of dependency path are utilized. i.e., these words and their associated constituents are added.*

*The results show that SDP-CPT can achieve the best performance.*
Experimental Results 2

- Contribution of different kinds of dependencies on the AIMed corpus

<table>
<thead>
<tr>
<th>Types</th>
<th>P(%)</th>
<th>R(%)</th>
<th>F1</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCP</td>
<td>45.0</td>
<td>19.5</td>
<td>26.5</td>
<td>67.9</td>
</tr>
<tr>
<td>Argument</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+subj</td>
<td>52.5</td>
<td>33.2</td>
<td>40.4</td>
<td>72.7</td>
</tr>
<tr>
<td>+obj</td>
<td>56.2</td>
<td>46.2</td>
<td>50.4</td>
<td>76.6</td>
</tr>
<tr>
<td>+arg-others</td>
<td>56.1</td>
<td>47.0</td>
<td>50.9</td>
<td>76.5</td>
</tr>
<tr>
<td>Modifier</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+nn</td>
<td>58.1</td>
<td>53.4</td>
<td>55.1</td>
<td>81.4</td>
</tr>
<tr>
<td>+prep</td>
<td>58.2</td>
<td>55.2</td>
<td>56.6</td>
<td>83.1</td>
</tr>
<tr>
<td>+mod-others</td>
<td>59.1</td>
<td>57.6</td>
<td>58.1</td>
<td>83.3</td>
</tr>
<tr>
<td>Conj</td>
<td>58.9</td>
<td>55.0</td>
<td>56.7</td>
<td>82.8</td>
</tr>
<tr>
<td>Others</td>
<td>58.4</td>
<td>53.8</td>
<td>55.8</td>
<td>83.0</td>
</tr>
</tbody>
</table>

- Different kinds of dependencies are used accumulatively.
- *Subj, obj, prep and nn yield substantial performance improvements.*
- *Conj and Others harm the performance.*
Experimental Results 3

- Comparison of different constituent parse tree structures across major PPI corpora

<table>
<thead>
<tr>
<th>Tree Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCT</td>
<td>Minimum Complete Subtree</td>
</tr>
<tr>
<td>SPT</td>
<td>Shortest Path-enclosed Tree</td>
</tr>
<tr>
<td>CS-SPT</td>
<td>Context-Sensitive Shortest Path-enclosed Tree</td>
</tr>
<tr>
<td>DSPT</td>
<td>Dynamic Syntactic Parse Tree</td>
</tr>
<tr>
<td>SDP-CPT</td>
<td>Shortest Dependency Path- Constituent Parse Tree</td>
</tr>
</tbody>
</table>

SDP-CPT performs best and outperforms SPT consistently on most PPI corpora;
**Experimental Results 4**

- Comparison of kernel-based PPI extraction systems on the AIMed corpus

<table>
<thead>
<tr>
<th>Single kernel PPI</th>
<th>P(%)</th>
<th>R(%)</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our SDP-CPT</td>
<td>59.7</td>
<td>56.9</td>
<td>58.1</td>
</tr>
<tr>
<td>Kim et al. (2010)</td>
<td>61.4</td>
<td>53.3</td>
<td>56.7</td>
</tr>
<tr>
<td>Airola et al. (2008)</td>
<td>52.9</td>
<td>61.8</td>
<td>56.4</td>
</tr>
<tr>
<td>Bunescu et al. (2005)</td>
<td>65.0</td>
<td>46.4</td>
<td>54.2</td>
</tr>
<tr>
<td>Tikk et al. (2010)</td>
<td>39.2</td>
<td>31.9</td>
<td>34.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Multiple kernel PPI</th>
<th>P(%)</th>
<th>R(%)</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sætre et al. (2007)</td>
<td>64.3</td>
<td>44.1</td>
<td>52.0</td>
</tr>
<tr>
<td>(BOW+Dependency path)</td>
<td>50.9</td>
<td>56.1</td>
<td>53.0</td>
</tr>
<tr>
<td>Miyao et al. (2008)</td>
<td>60.9</td>
<td>57.2</td>
<td>59.0</td>
</tr>
<tr>
<td>(BOW+Constituent parse tree)</td>
<td>54.9</td>
<td>65.5</td>
<td>59.5</td>
</tr>
<tr>
<td>Giuliano et al. (2006)</td>
<td>-</td>
<td>-</td>
<td>64.2</td>
</tr>
<tr>
<td>Miyao et al. (2008)</td>
<td>-</td>
<td>-</td>
<td>54.9</td>
</tr>
<tr>
<td>(Dependency+PAS)</td>
<td>-</td>
<td>-</td>
<td>65.5</td>
</tr>
<tr>
<td>Miwa et al. (2009)</td>
<td>-</td>
<td>-</td>
<td>59.5</td>
</tr>
<tr>
<td>(BOW+Shortest path+Dependency graph)</td>
<td>-</td>
<td>-</td>
<td>64.2</td>
</tr>
</tbody>
</table>

Our SDP-CPT performs best among single kernel PPI systems with the balanced Precision and Recall, though it performs worse than some multiple kernel PPI systems.
Conclusion and future work

★ Conclusion

• Constituent parse tree kernel can achieve promising results for PPI extraction with the tree trimmed according to dependency information.

• our dependency-directed constituent parse tree structure provides a general way to automatically determine the constituent parse tree for a wide class of related learning tasks, such as RE, SRL or even coreference resolution.

★ Future work

• Further investigate the synergy between dependency and constituent-based syntactic information.
Reference

Comments

Thank you for your attention!