Dependency Parsing: Past, Present, and Future

Wenliang Chen, Zhenghua Li, Min Zhang
\{wlchen, zhlili13, minzhang\}@suda.edu.cn
Soochow University, China
Recent Events of Dependency Parsing

- Shared tasks
  - CoNLL 2006/2007 shared tasks on multilingual dependency parsing
  - CoNLL 2008/2009 shared tasks on joint parsing of syntactic and semantic dependencies
  - SANCL 2012 shared task organized by Google (parsing the web)
  - SemEval 2014/2015: broad-coverage semantic dependency parsing (SDP)
Recent Events of Dependency Parsing

- **Tutorials**
  - COLING-ACL06: "Dependency Parsing" by Joakim Nivre and Sandra Kubler
  - NAACL10: "Recent Advances in Dependency Parsing" by Qin Iris Wang and Yue Zhang
  - IJCNLP13: Ours
  - EACL14: "Recent Advances in Dependency Parsing" by Ryan McDonald and Joakim Nivre
  - ACL14: "Syntactic Processing Using Global Discriminative Learning and Beam-Search Decoding" by Yue Zhang, Meishan Zhang, and Ting Liu

- **Books**
  - "Dependency Parsing" by Sandra Kübler, Joakim Nivre, and Ryan McDonald, 2009
Outline

- Part A: dependency parsing and supervised approaches
- Part B: semi-supervised dependency parsing
- Part C: Parsing the web and domain adaptation
- Part D: Multilingual dependency parsing
- Part E: Conclusion and open problems
Part A: Dependency Parsing and Supervised approaches
A dependency tree is a tree structure composed of the input words and meets a few constraints:

- Single-head
- Connected
- Acyclic
Projective dependency trees

- Informally, “projective” means the tree does not contain any crossing arcs.
Non-projective dependency trees

- A non-projective dependency tree contains crossing arcs.

Example from “Dependency Parsing” by Kübler, Nivre, and McDonald, 2009
Dependency Tree

- The basic unit is a link (dependency, arc) from the head to the modifier.

Diagram:

- **obj**
- **eat**
  - Head
  - Governor
  - Parent
- **fish**
  - Modifier
  - Dependent
  - Child
- **eat** connects to **obj**
- **fish** connects to **obj**

- **Label**
- **Relation**
- **Type**
Dependency Tree

- A bilingual example
Evaluation Metrics

- Unlabeled attachment score (UAS)
  - The percent of words that have the correct heads

- Labeled attachment score (LAS)
  - The percent of words that have the correct heads and labels.

- Root Accuracy (RA)

- Complete Match rate (CM)
Formalism of Dependency Parsing

\[ Y^* = \arg \max_{Y \in \Phi(X)} \text{score}(X, Y) \]

\( X = x_1, x_2, \ldots, x_n \rightarrow \) the input sentence

\((h, m) \rightarrow \) a link from the head \( x_h \) to the modifier \( x_m \)

\( Y = \{(h, m) : 0 \leq h \leq n, 0 < m \leq n\} \rightarrow \) a candidate tree

\( \Phi(X) \rightarrow \) The set of all possible dependency trees over \( X \)
Supervised Approaches for Dependency Parsing

- Graph-based
- Transition-based
- Hybrid (ensemble)
- Other methods
Graph-based Dependency Parsing

- Find the highest scoring tree from a complete dependency graph.

\[ Y^* = \arg \max_{Y \in \Phi(X)} \text{score}(X, Y) \]
Two problems

- The search problem
  - Given the score of each link, how to find $Y^*$?

- The learning problem
  - Given an input sentence, how to determine the score of each link? $w \cdot f$
  - How to learn $w$ using a treebank (supervised learning)?

$$Y^* = \arg \max_{Y \in \Phi(X)} \text{score}(X, Y)$$
First-order as an example

- The first-order graph-based method assumes that the dependencies in a tree are independent from each other (arc-factorization)

\[
score(X, Y) = \sum_{(h, m) \in Y} score(X, h, m)
\]
Search problem for first-order model

- Eisner (2000) described a **dynamic programming** based decoding algorithm for bilexical grammar.
- McDonald+ (2005) applied this algorithm to the search problem of the first-order model.
- Time complexity $O(n^3)$
Eisner algorithm data structure

- Basic data structure
  - Incomplete spans
Eisner algorithm data structure

- Basic data structure
  - Complete spans
Eisner algorithm operations

- Basic Operations

\[
\begin{align*}
s & \quad r & \quad r+1 & \quad t \\
\triangle & + & \triangle & \rightarrow \\
\begin{cases}
s & \quad t \\
\triangle & + & \triangle & \rightarrow \\
\triangle & + & \triangle & \rightarrow 
\end{cases}
\end{align*}
\]
Eisner algorithm

- Initialization: complete spans (width=1)
Eisner algorithm

- Build incomplete span (width = 2)
Eisner algorithm

- Build complete spans (width = 2)
Eisner algorithm

- Build incomplete span (width = 3)

? \( \text{score(does} \rightarrow \text{here)} = 3.3 \)
Eisner algorithm

- Build incomplete spans (width = 3)

\[
\begin{align*}
\text{does_2} + \text{it}_3 & \quad \Rightarrow \quad \text{does_2} + \text{it}_3, \quad \text{score}(\text{does} \rightarrow \text{here}) = 3.3 \\
\text{does_2} + \text{it}_3 & \quad \Rightarrow \quad \text{does_2} + \text{it}_3, \quad \text{score}(\text{does} \rightarrow \text{here}) = 3.3 \\
\end{align*}
\]
Eisner algorithm

- Build complete spans (width = 3)
- Build incomplete spans (width = 4)
- ...
- The best parse is stored in the complete span from “$” to “here”: C(0,4)
Eisner algorithm

- The search path for the correct parse tree

Complete!
The learning problem

- Given an input sentence, how to determine the score of each link?

\[ \text{score}(2,4) = ? \]

- Feature based representation: a link is represented as a feature vector \( \mathbf{f}(2,4) \)

\[ \text{score}(2,4) = \mathbf{w} \cdot \mathbf{f}(2,4) \]
Features for one dependency

Example from slides of Rush and Petrov (2012)
How to learn $w$? (supervised)

- Use a treebank
  - Each sentence has a manually annotated dependency tree.
- Online training (Collins, 2002; Crammer and Singer, 2001; Crammer+, 2003)
  - Initialize $w = 0$
  - Go though the treebank for a few (10) iterations.
    - Use one instance to update the weight vector.
Online learning $w$

Gold-standard parse $Y^+$

$$w^{(k+1)} = w^{(k)} + f(X,Y^+) - f(X,Y^-)$$

1-best parse $Y^-$ with $w^{(k)}$
Quick summarization

• The search problem
  • Dynamic programming based decoding algorithms (Eisner algorithm) to find the best parse tree.

• The learning problem
  • Online training algorithms to learn the feature weight vector $w$ under the supervision of a treebank.
Recent advances (graph-based method)

- Explore features of more non-local subtrees
  - Second-order (McDonald & Pereira 06; Carreras 07)
  - Third-order (Koo+, 2010)
  - Higher-order with beam search (Zhang & McDonald 12)
Transition-based Dependency Parsing

- Gradually build a tree by applying a sequence of transition actions. (Yamada and Matsumoto, 2003; Nivre, 2003)
- The score of the tree is equal to the summation of the scores of the actions.

\[
    score(X, Y) = \sum_{i=0}^{m} score(X, h_i, a_i)
\]

- \(a_i\) → the action adopted in step \(i\)
- \(h_i\) → the partial results built so far by \(a_0...a_{i-1}\)
- \(Y\) → the tree built by the action sequence \(a_0...a_m\)
Transition-based Dependency Parsing

The goal of a transition-based dependency parser is to find the highest scoring action sequence that builds a legal tree.

\[
Y^* = \arg \max_{Y \in \Phi(X)} \text{score}(X, Y)
\]

\[
= \arg \max \sum_{a_0 \ldots a_m \rightarrow Y}^m \text{score}(X, h_i, a_i)
\]
Two problems for Transition-based DP

- The search problem
  - Assuming the model parameters (feature weights) are known, how to find the optimal action sequence that leads to a legal tree for an input sentence?
  - Greedy search as an example

- The learning problem
  - How to learn the feature weights?
  - Global online training
    - Before 2008: Locally training classifiers
Components of Transition-based DP

- A stack to store the processed words and the partial trees
- A queue to store the unseen input words
- Transition actions
  - Gradually build a dependency tree according to the contexts (history) in the stack and the queue.

Which action should be applied?
An arc-eager transition-based parser

- Four actions
  - Shift
  - Left-arc
  - Right-arc
  - Reduce
An arc-eager transition-based parser

Stack

$0$ He$_1$

Input queue

He$_1$ does$_2$ it$_3$ here$_4$

↓ Shift

$0$ He$_1$

does$_2$ it$_3$ here$_4$
An arc-eager transition-based parser

$0 \quad \text{He}_1 \quad \text{does}_2 \quad \text{it}_3 \quad \text{here}_4$

$0 \quad \text{does}_2 \quad \text{it}_3 \quad \text{here}_4$

$\text{He}_1$
An arc-eager transition-based parser

$0 \text{ does}_2 \text{ it}_3 \text{ here}_4

\text{He}_1 \text{ Right-arc}

$0 \text{ does}_2 \text{ it}_3 \text{ here}_4

\text{He}_1
An arc-eager transition-based parser

He$_1$ does$_2$ it$_3$ here$_4$

Right-arc

He$_1$

$0$ does$_2$ it$_3$ here$_4$
An arc-eager transition-based parser

Reduce

He₁
does₂
it₃
gerе₄

He₁

does₂

He₁

it₃
gerе₄
An arc-eager transition-based parser

$0 \quad \text{does}_2 \quad \text{He}_1 \quad \text{it}_3$

$0 \quad \text{does}_2 \quad \text{here}_4 \quad \text{H}_e_1 \quad \text{it}_3$

Right-arc
An arc-eager transition-based parser

$0 \quad \text{does}_2 \quad \text{here}_4$

$\text{He}_1 \quad \text{it}_3$

Reduce

$0 \quad \text{does}_2$

$\text{He}_1 \quad \text{it}_3 \quad \text{here}_4$
An arc-eager transition-based parser

Reduce

Complete!
Recent advances (transition-based)

- Explore richer features using the partially built structures (Zhang and Nivre, 2011).
- Enlarge the search space during decoding
  - Beam search (Duan+, 2007; Zhang and Nivre, 2011)
  - Beam search with state merging via dynamic programming (Huang and Sagae, 2010)
  - Dynamic programming based decoding (Kuhlmann+, 2011)
- Better online learning
  - Dynamic oracles (Goldberg and Nivre, 2014)
Other methods

- Easy-first non-directional dependency parsing (Goldberg and Elhadad, 2010)
  - Iteratively select easiest (highest-scoring) pair of neighbors to build a dependency.

- Constituent-based dependency parsing (Sun and Wan, 2013)
  - Method 1: convert the outputs of a constituent parser into dependency structures.
  - Method 2: convert dependency trees into context-free-grammar structures.
Ensemble methods

• Different dependency parsers have different advantages.
  • The graph-based MSTParser performs better on long-distance dependencies.
  • The transition-based MaltParser performs better on short-distance dependencies.

• Ensemble methods try to leverage the complementary strengths of different parsing approaches.
  • Re-parsing (Sagae and Lavie, 2006; Surdeanu and Manning, 2010)
  • Stacking (McDonald and Nivre, 2011; Martins+, 2008)
Ensemble via re-parsing

- Sagae and Lavie (2006); Surdeanu and Manning (2010)
  - Separately train M different parsers
  - For a test sentence, the M parsers produce M parses.
  - Combine the M parses to build a partial dependency graph.
  - Reparse to find the best result from the dependency graph using the standard MST parsing algorithm.
Ensemble via re-parsing

- Sagae and Lavie (2006); Surdeanu and Manning (2010)

Example from the slides of Wang and Zhang (2010)
Ensemble via stacking

- Joakim and McDonald (2008); Martins+ (2008)
  - Combine the graph-based and transition-based parsers.
  - Use one parser to guide or help the other one.
    - Train the level-1 parser first (Parser1)
    - Let the level-2 parser (Parser2) consult Parser1 during both training and test.
  - Two directions
    - $\text{MST}_{\text{malt}}$ (MaltParser for level-1; MSTParser as level-2)
    - $\text{Malt}_{\text{MST}}$ (verse)
Non-projective dependency parsing

- Pseudo-projective (Nivre and Nilsson, 2005)
- Graph-based methods (McDonald+, 2005, 2006; Pilter, 2014)
- Transition-based methods (Nivre, 2009)

Example from “Dependency Parsing” by Kübler, Nivre, and McDonald, 2009
Non-projective dependency parsing

- Non-projectivity in natural languages

<table>
<thead>
<tr>
<th>Language</th>
<th>Trees</th>
<th>Arcs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>11.2%</td>
<td>0.4%</td>
</tr>
<tr>
<td>Basque</td>
<td>26.2%</td>
<td>2.9%</td>
</tr>
<tr>
<td>Czech</td>
<td>23.2%</td>
<td>1.9%</td>
</tr>
<tr>
<td>Danish</td>
<td>15.6%</td>
<td>1.0%</td>
</tr>
<tr>
<td>Greek</td>
<td>20.3%</td>
<td>1.1%</td>
</tr>
<tr>
<td>Russian</td>
<td>10.6%</td>
<td>0.9%</td>
</tr>
<tr>
<td>Slovene</td>
<td>22.2%</td>
<td>1.9%</td>
</tr>
<tr>
<td>Turkish</td>
<td>11.6%</td>
<td>1.5%</td>
</tr>
</tbody>
</table>

Table from invited talks by Prof. Joakim Niver: [Beyond MaltParser - Advances in Transition-Based Dependency Parsing](#)
Non-projective dependency parsing

- Pseudo-projective (Nivre and Nilsson, 2005)
  - Pre-processing
    - Convert non-projective trees into projective ones, with dependency labels encoding the transform process.
  - Projective dependency parsing for both training and test.
- Post-processing
  - Recover the non-projective trees from the projective outputs with the help of the labels.
Non-projective dependency parsing

- Graph-based methods
  - First-order (McDonald+, 2005)
    - Chu-Liu-Edmond decoding algorithm, $O(n^2)$
  - Second-order (McDonald and Pereira, 2006)
    - Greedy approximate decoding, $O(n^3)$
  - Third-order (Pitler, 2014)
    - Dynamic programming decoding, $O(n^4)$
Non-projective dependency parsing

- Transition-based methods
- Extended with a SWAP action (Nivre, 2009)

Example from Wang and Zhang (2010)
Probabilistic dependency parsing

- Log-linear models (CRF)
  - Projective: inside-outside (Paskin, 2001)
  - Non-projective: matrix-tree theorem (Smith and Smith, 2007; McDonald and Satta, 2007; Koo+, 2007)

- Generative models
  - Spectral learning (Luque+, 2012; Dhillon+, 2012)
Improving parsing efficiency

- Coarse-to-fine parsing
- Fast feature generation
- Parallel techniques
Coarse-to-fine parsing

• Use a coarse and fast model to prune the search space for more complex models.
  • Charniak and Johnson (2005) apply this method to fast n-best constituent parsing.
  • Use a CRF-based first-order dependency parser to prune the search space before third-order parsing (Koo and Collins, 2010)
• Vine pruning (Rush and Petrov, 2012)
  • Zero-order => first-order => second-order => third-order
Fast feature generation

- When only considering positive features, current graph-based dependency parsers usually incorporate tens of millions of different features.
- Feature generation based on standard hash tables needs ~90% of the total parsing time (Bohnet, 2010)
  - Feature string generation
  - Feature index mapping
Feature strings for one dependency

Example from slides of Rush and Petrov (2012)
Feature-index mapping

• Map different feature strings into different indices (feature space).

\[
\begin{align*}
\text{[went, VERB, As, IN]} & \rightarrow 1023 \\
\text{[VERB, As, IN, left]} & \rightarrow 526
\end{align*}
\]

• Feature weight
  • Each feature in the feature space has a weight.
Fast feature generation

  - Map each feature string into an index using a hash function with pure numerical calculation.
  - Hash collision is OK.
- Qian + (2010) propose a 2D trie structure for fast feature generation.
  - Complex feature templates are extension of simple ones.
Parallel techniques

• Parsing efficiency can be largely improved by exploring multiple CPU cores via multi-thread programming.
  • Graph-based parser (Bohnet, 2010)
    • Parallel feature generation
    • Parallel decoding algorithms
  • Transition-based parser (Hatori+, 2011)
    • Parallel beam-search decoding
• Both parsers are publicly available.
Quick summarization

Supervised dependency parsing

Graph-based

Transition-based

Easy-first constituent-based

Non-projective

Probabilistic models

Parsing efficiency

Parser ensemble: reparsing, stacking
Joint morphological analysis and dependency parsing

• Motivation
  • Due to the intrinsic difficulty of NLP, a cascaded framework is commonly adopted.
    • Morphology (lexicon) → Syntax → Semantics
  
• Two problems
  • Error propagation
  • Fail to explore the connections between different levels of processing tasks which may help those related tasks.
Pipeline example: Chinese POS tagging and dependency parsing

Dependency Parsing

POS tagging

Words

Owen now plays for Liverpool.

欧文现在效力于利物浦队。
Joint Chinese POS tagging and dependency parsing

Joint Parsing and Tagging

POS tag lattice

Words

$0$ 欧文 现在 效力 于 利物浦队 .

Owen now palys for in Liverpool .
Formally, the pipeline method

Step 1: POS tagging

$$T^* = \arg \max_{T \in \Phi_1(X)} \text{score}_{pos}(X, T)$$

Step 2: dependency parsing

$$Y^* = \arg \max_{Y \in \Phi_2(X)} \text{score}_{syn}(X, T^*, Y)$$
Graph-based joint POS tagging and dependency parsing (Li+, 2011)

• The joint method tries to solve the two tasks simultaneously.

\[
\langle T^*, Y^* \rangle = \arg \max_{T \in \Phi_1(X), Y \in \Phi_2(X)} \text{score}_{\text{joint}}(X, T, Y)
\]
Graph-based joint POS tagging and dependency parsing (Li+, 2011)

\[
\begin{align*}
\text{score}_{\text{joint}}(X, T, Y) &= \text{score}_{\text{pos}}(X, T) + \text{score}_{\text{syn}}(X, T, Y) \\
&= w_{\text{pos}} \cdot f_{\text{pos}}(X, T) + w_{\text{syn}} \cdot f_{\text{syn}}(X, T, Y) \\
&= w_{\text{pos} \oplus \text{syn}} \cdot f_{\text{pos} \oplus \text{syn}}(X, T, Y) \\
&= w_{\text{joint}} \cdot f_{\text{joint}}(X, T, Y)
\end{align*}
\]

Under the joint model, the POS tagging features and the syntactic features can interact with each other in order to find an optimal joint solution.
Graph-based joint POS tagging and dependency parsing (Li+, 2011)

- The search problem
  - Given the feature weights $w_{\text{joint}}$, how to efficiently find the optimal joint result from a huge search space?
  - Dynamic programming based decoding algorithms: direct extension of the decoding algorithms for dependency parsing
Dynamic programming based decoding algorithms (Li+, 2011)

- Product of two dynamic programming based decoding algorithms
- Augment partial parses (spans) with POS tags.
- Time complexity $O(n^3q^4)$ ($q=1.4$)
The learning problem (Li+, 2011)

- How to learn the feature weights $w_{\text{joint}}$?
- Online training
  - Averaged perceptron (AP)
  - Margin infused relaxed algorithm (MIRA)
  - Passive-aggressive algorithm (PA)
Separately passive-aggressive (SPA) learning (Li+, 2012)

- Use separate update steps for the POS tagging features and syntactic features.
- Can better balance the discriminative power of both tagging and parsing features.
- Lead to better tagging and parsing accuracy.
Results of graph-based joint models

On Chinese Data: CTB5

**POS tagging accuracy**
Transition-based joint Chinese POS tagging and dependency parsing

- A direct extension of transition-based dependency parsing by adding a tagging action (Hatori+, 2011; Bohnet and Nivre, 2012)

- An arc-standard version (Hatori+, 2011)
  - Shift($t$): shift a word in the queue into the stack and assign tag $t$ to it. (SH)
  - Reduce-left (RL)
  - Reduce-right (RR)
### Transition-based joint Chinese POS tagging and dependency parsing

- An example from the slides of Hatori+ (2011)

<table>
<thead>
<tr>
<th>#</th>
<th>Act.</th>
<th>Stack S</th>
<th>Queue Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>−</td>
<td>φ</td>
<td>我/?? 想/?? 把/?? …</td>
</tr>
<tr>
<td>1</td>
<td>SH(PN)</td>
<td>我/PN</td>
<td>想/?? 把/?? 这/?? …</td>
</tr>
<tr>
<td>2</td>
<td>SH(VV)</td>
<td>我/PN 想/VV</td>
<td>把/?? 这/?? 个/?? …</td>
</tr>
<tr>
<td>3</td>
<td>SH(BA)</td>
<td>我/PN 想/VV 把/BA</td>
<td>这/?? 个/?? 句子/?? …</td>
</tr>
<tr>
<td>4</td>
<td>SH(DT)</td>
<td>我/PN 想/VV 把/BA 这/DT</td>
<td>个/?? 句子/?? 翻译/?? …</td>
</tr>
<tr>
<td>5</td>
<td>SH(M)</td>
<td>我/PN 想/VV 把/BA 这/DT 个/M</td>
<td>句子/?? 翻译/?? 成/?? …</td>
</tr>
<tr>
<td>6</td>
<td>RL</td>
<td>我/PN 想/VV 把/BA 这/DT〜[个/M]</td>
<td>句子/?? 翻译/?? 成/?? …</td>
</tr>
<tr>
<td>7</td>
<td>SH(NN)</td>
<td>我/PN 想/VV 把/BA 这/DT〜[个/M] 句子/NN</td>
<td>翻译/?? 成/?? 英语/?? $</td>
</tr>
<tr>
<td>8</td>
<td>RR</td>
<td>我/PN 想/VV 把/BA [这/DT〜[…]]〜句子/NN</td>
<td>翻译/?? 成/?? 英语/?? $</td>
</tr>
<tr>
<td>9</td>
<td>RL</td>
<td>我/PN 想/VV 把/BA〜[…]〜句子/NN</td>
<td>翻译/?? 成/?? 英语/?? $</td>
</tr>
<tr>
<td>10</td>
<td>SH(VV)</td>
<td>我/PN 想/VV 把/BA〜[…]〜句子/NN 翻译/VV</td>
<td>成/?? 英语/?? $</td>
</tr>
</tbody>
</table>
Other work on joint morphological analysis and parsing

- Easy-first joint Chinese POS tagging and dependency parsing (Ma+, 2012)
- Transition-based joint Chinese word segmentation, POS tagging, and dependency parsing (Hatori+, 2012; Li and Zhou, 2012)
- Joint Chinese word segmentation, POS tagging, and phrase-structure parsing
  - Zhang+ (2013): a character-level transition-based system
Other work on joint morphological analysis and parsing

- **Dual decomposition (DD) (Rush+, 2010)**
  - Integrating different NLP subtasks at the test phase
    - a phrase-structure parser and a dependency parser
    - a phrase-structure parser and a POS tagger

- **Loopy belief propagation (LBP) (Lee+, 2011)**
  - Joint morphological disambiguation and dependency parsing for morphologically-rich languages including Latin, Czech, Ancient Greek, and Hungarian.

- **Comparison of DD and LBP (Auli and Lopez, 2011)**
  - Joint CCG supertagging and parsing
References

References

References

References

- Yoav Goldberg and Joakim Nivre. 2014. Training Deterministic Parsers with Non-Deterministic Oracles. In TACL.
References

- Emily Pitler. 2014. A Crossing-Sensitive Third-Order Factorization for Dependency Parsing. In TACL
- Xian Qian, Qi Zhang, Xuangjing Huang, Lide Wu. 2010. 2D Trie for Fast Parsing. In Proc. of COLING, pp904-912
References

End of Part A
Part B: Semi-supervised dependency parsing for in-domain texts
Semi-supervised dependency parsing

- Supervised parsing
  - Training: Labeled data
- Semi-supervised parsing
  - Training: Additional unlabeled data + labeled data
In-domain vs out-domain

Annotated data in Domain A

Training

A Parser

Parsing texts in Domain A

Parsing texts in Domain B

In-domain

Out-domain
Semi-supervised dependency parsing

- Typical methods
  - Use whole auto-parsed trees
    - Self-training
    - Co-training

- Other methods?
  - To use partial trees
Semi-supervised dependency parsing

Unlabeled

He ate the fish
The book is mine
Put it there

... →

Lexical Level
Word clusters, word co-occurrences ...

Parsing

Auto-parsed trees

He ate the fish
The book is mine
Put it there

... →

Partial Tree Level
Subtree frequencies, subtree likelihood

→

Whole Tree Level
Reliable whole trees as extra training instances
Semi-supervised dependency parsing

- Whole tree level
- Partial tree level
- Lexical level
-Whole tree level

• Approaches:
  • Self-training
    • Use one parser
    • Select the automatic parses from unlabeled data as extra training examples
  • Co-training
    • Use two parsers
    • Select the automatic parses from two different views from unlabeled data as extra training examples
  • Ambiguity-aware ensemble training
    • Aggregate multi parsers’ outputs into forests
Self-training

- Self-training
  - Step 1: Train a first-stage parser with the human labeled data
  - Step 2: Apply the parser to produce automatic parses for the unlabeled data
  - Step 3: Select some auto-parsed sentences as newly labeled data (How to select)
  - Step 4: Train a better parser by combining the human labeled and selected newly auto-parsed data (How to combine).
Self-training (McClosky et al., 2006)

- Self-training with a two-stage constituent parser
  - Step 1: train a first-stage generative parser and a second-stage discriminative reranker with the labeled data (Charniak and Johnson, 2005)
  - Step 2: apply the parser and reranker to produce automatic parses for the unlabeled data.
  - Step 3: train a better first-stage parser by combining the labeled and unlabeled data (with corpus weighting).
Self-training (Huang and Harper, 2009)

- Self-training with a single PCFG-LA parser
  - Step 1: train a Berkeley parser with the labeled data (Petrov and Klein, 2007)
  - Step 2: apply the parser produce automatic parses for the unlabeled data.
  - Step 3: train a better parser by combining the labeled and unlabeled data (with corpus weighting).
Self-training (Huang et al., 2010)

- Self-training with products of PCFG-LA grammars
  - Use the products of PCFG-LA parsers to
    - alleviate the problem that EM tends to get stuck in local maxima;
    - produce better parses for the unlabeled data.
Self-training (dependency parsing)

- For dependency parsing (in-domain)
  - Hard to pick up reliable sentences
    - Use SVM classifier (Kawahara and Uchimoto, 2008) to select reliable sentences
  - Not so successful (so far)
Co-training

- Co-training with two parsers of different views
  (Sarkar, 2001; Steedman et al., 2003; Sagae and Tsujii, 2007)
  - Step 1: select two different parsers, e.g., graph-based and transition-based dependency parsers.
  - Step 2: train the two parsers with the labeled data.
  - Step 3: apply the two parsers to the pool of raw sentences.
  - Step 4: select the reliable parses according to the consistency between the two parsers, and add them into the labeled data.
  - Go to Step 2 until no performance improvement on some held-out data.
Ambiguity-aware ensemble training (Li+ 14)

- Use different parsers to parse the unlabeled data, and aggregate their outputs into forests;
- Use the unlabeled data with forest as extra training data.
Ambiguity-aware ensemble training (Li+ 14)

- Training objective

\[ \mathcal{L}(D'; w) = \sum_{i=1}^{M} \log \left( \sum_{d' \in \mathcal{V}_i} p(d'|u_i; w) \right) \]
Ambiguity-aware ensemble training (Li+ 14)

- Experimental results show
  - Better than co-training/tri-training
  - Diversity of parsers is important!
    - Generative constituent parser is the most useful
Semi-supervised dependency parsing

- Whole tree level
- Partial tree level
- Lexical level
- Partial tree level

  • Approaches
    • Use word pairs (Noord, 2007; Chen et al., 2008)
    • Use subtrees (Chen et al., 2009)/DLM (Chen et al., 2012)
    • Use hybrid discriminative and generative models to derive useful cues from unlabeled data (Suzuki et al., 2009).
    • Use meta features (Chen et al., 2013)
- Partial tree level

- With word pairs (Noord, 2007; Chen et.al. 2008)
Word pairs (Noord, 2007; Chen et.al. 2008)

- Parsing with word pairs
  - Step1: Train a baseline parser with the labeled data
  - Step2: Use the baseline parser to parse the raw sentences
  - Step3: Collect word pairs (lexical dependencies)
  - Step4: Represent new features based on word pairs
  - Step5: Re-train a new parser by combining the base features and new features
Word pairs

- Statistics of word pairs
  - Pointwise mutual information (Noord, 2007)
    \[
    I(r(w_1, w_2)) = \log \frac{f(r(w_1, w_2))}{f(r(w_1, -)) f(-, w_2)}
    \]
  - Association score between w1 and w2
Word pairs

- Statistics of word pairs
  - Short dependency (Chen et. al. 2008)
    - Word pairs with dependency length=1/2
    - Group the collected word pairs into buckets
- Partial tree level

- With word pairs
  - The information of word pairs is too few.

- With subtrees
With subtrees

• Parsing with subtrees
  • Step1: Train a baseline parser with the labeled data
  • Step2: Use the baseline parser to parse the raw sentences
  • Step3: Extract subtrees from the auto-parsed data
  • Step4: Represent new features based on the extracted subtrees
  • Step5: Re-train a new parser by combining the base features and subtree-based features
Subtree Extraction

- Extract subtrees containing two nodes or three nodes
  - If a subtree contains two nodes, we call it a bigram-subtree.
  - If a subtree contains three nodes, we call it a trigram-subtree.
Subtree Extraction

```
<table>
<thead>
<tr>
<th>ate</th>
<th>l:1:1-ate:2:0</th>
<th>ate</th>
<th>ate:1:0-with:2:1</th>
<th>fish</th>
<th>the:1:1-fish:2:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>fish</td>
<td>ate:1:0-fish:2:1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fork</td>
<td>a:1:1-fork:2:0</td>
<td>with</td>
<td>with:1:0-fork:2:1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td>fork</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Diagram:
- Root node `ROOT` with children `ate`, `fish`, and `with`.
- `ate` has children `l` and `the`.
- `fish` has children `the` and `fork`.
- `with` has children `a` and `fork`.
### Subtree Extraction

**Diagram:***

- **ROOT**
- **ate**
- **fish**
- **with**
- **a**
- **fork**

---

<table>
<thead>
<tr>
<th>ate</th>
<th>fish</th>
<th>with</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ate:1:0-fish:2:1</td>
<td></td>
</tr>
<tr>
<td>ate</td>
<td>with</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ate:1:0-with:2:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fish</td>
<td>the:1:1-fish:2:0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>fork</th>
<th>a:1:1-fork:2:0</th>
<th>with</th>
</tr>
</thead>
<tbody>
<tr>
<td>fish</td>
<td>ate:1:0-fish:2:1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>with:1:0-fork:2:1</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>ate:1:0-with:2:1</td>
<td></td>
</tr>
<tr>
<td>with</td>
<td>ate:1:0-with:2:1</td>
<td></td>
</tr>
</tbody>
</table>

---

**Example:***

- ate:1:0-fish:2:1-with:3:1
- ate:1:0-with:2:1-..:3:1
Subtree Labeling

- To group the extracted subtrees into different sets according to their frequencies

- Subtree labels:
  - After testing on development data, subtrees are grouped into three sets: HF, MF, and LF
  - Add new set (ZERO) to refer to the subtrees that are not included
  - Finally, we have four labels: HF, MF, LF, and ZERO
Feature representation

• Parsing model
  • A graph-based MST Parsing model proposed by McDonald et al. (2005) and McDonald and Pereira (2006)
    • First-order features are defined over single graph edges
    • Second-order features are defined on adjacent edges

• Represent two types of features on labeled training data
  • First-order features based on bigram-subtrees
  • Second-order features based on trigram-subtrees
First-order subtree-based features

- Generate the features for a head $h$ and a dependent $d$

- The links form temporary bigram-subtrees

- Subtree-based Features
  - The labels of these bigram-subtrees
  - The labels are conjoined with POS tags of head
  - The labels are conjoined with word forms of head
Second-order subtree-based features

- Generate the features for a head $h$ and a dependent $d_1$, $d_1$’s right-leftmost sibing $d_2$

- The links form temporary trigram-subtrees

Subtree-based Features
- The labels of these trigram-subtrees
- The labels are conjoined with POS tags of head
- The labels are conjoined with word forms of head
How does this approach work?

Training data
... ate ... fish ...

Auto-parsed data
... ate ... fish ...
... ate ... nut ...

Test data
... ate ... nut ...

Feature: MF
unseen

Additional features for unseen tuples
Results from Chen et al. 2009

- On PTB

<table>
<thead>
<tr>
<th></th>
<th>UAS</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ord1</td>
<td>90.95</td>
<td>37.45</td>
</tr>
<tr>
<td>Ord1s</td>
<td>91.76(+0.81)</td>
<td>40.68</td>
</tr>
<tr>
<td>Ord2</td>
<td>91.71</td>
<td>42.88</td>
</tr>
<tr>
<td>Ord2s</td>
<td>92.51(+0.80)</td>
<td>46.19</td>
</tr>
</tbody>
</table>
- Partial tree level

- With word pairs
  - The information of word pairs is too few.

- With subtrees

- With dependency language model (DLM)
Dependency Language Model

- **Standard Language Model (N-gram)**
  - Predicts the next word based on the N-1 immediate previous words
  - Can not capture long-distance word relations

- **Dependency Language Model (Shen et. al, 2008)**
  - Predicts the next child of a head based on the N-1 immediate previous children and the head itself
  - Can capture long-distance word relations
  - Has been used in SMT (Shen et. al, 2008)
Dependency Language Model

- Input sentence: \( x = (x_0, x_1, \ldots, x_i, \ldots, x_n) \)
- Dependency tree: \( y \)
- Set for the words having at least one dependent: \( H(y) \)
- Dependency structure with head \( x_h : D_h \)

\[
P(y) = \prod_{x_h \in H(y)} P(D_h)
\]
Dependency Language Model

- $D_h$

\[
P(D_h) = P_L(D_h) \times P_R(D_h) \quad (1)
\]

\[
P_L(D_h) \approx P_{L_c}(x_{L1}|x_h) \\
\times P_{L_c}(x_{L2}|x_{L1}, x_h) \\
\times \ldots \\
\times P_{L_c}(x_{Lk}|x_{L(k-1)}, \ldots, x_{L(k-N+1)}, x_h) 
\]

\[
P_R(D_h) \approx P_{R_c}(x_{R1}|x_h) \\
\times P_{R_c}(x_{R2}|x_{R1}, x_h) \\
\times \ldots \\
\times P_{R_c}(x_{Rk}|x_{R(k-1)}, \ldots, x_{R(k-N+1)}, x_h) 
\]
Dependency Language Model

\[ P(y) = P(D_{ate}) \times P(D_{meat}) \times P(D_{with}) \times P(D_{fork}) \]

\[ P(D_{ate}) = P_L(D_{ate}) \times P_R(D_{ate}) \]

\[ P_L(D_{ate}) = P_{Lc}(he|ate) \]

\[ P_R(D_{ate}) = P_{Rc}(with|ate) \times P_{Rc}(with|meat, ate) \]
Parsing with DLM

- Graph-based parsing model
- Add DLM scores
- DLM-based feature templates
Graph-based model

- Find a maximum spanning tree (MST) (McDonald et al., 2005)

\[ y^* = \arg \max_{y \in T(G_x)} s(x, y) = \arg \max_{y \in T(G_x)} \sum_{g \in y} \text{score}(\mathbf{w}, x, g) \]

\[ \text{score}(\mathbf{w}, x, g) = f(x, g) \cdot \mathbf{w} \]

\( f(x, g) \) is a high-dimensional feature representation which is based on arbitrary features of \( g \) and \( x \). \( \mathbf{w} \) is a weight vector.
Add DLM scores

- Consider the scores of DLM when searching for the MST

\[ y^*_{DLM} = \arg \max_{y \in T(G_x)} \left( \sum_{g \in y} \text{score}(w, x, g) + \text{score}^{DLM}(y) \right) \]

\[ s^* = \arg \max_{y \in T(G_x)} \sum_{g \in y} \text{score}(w, x, g) \]

\[ \text{score}^{DLM}(y) = f^{DLM} \cdot w^{DLM} \]

\[ P(y) = \prod_{x_h \in H(y)} P(D_h) \]
DLM-based feature templates

- Define DLM-based features for structure $D_h$.
  - $P_U(\text{ch})$: probability of generating left/right child ch
  - TYPE: left/right

\[
\Phi(P_U(\text{ch})) = \begin{cases} 
    PH & \text{if } No(P_U(\text{ch})) \leq \text{TOP10} \\
    PM & \text{if } \text{TOP10} < No(P_U(\text{ch})) \leq \text{TOP30} \\
    PL & \text{if } \text{TOP30} < No(P_U(\text{ch})) \\
    PO & \text{if } P_U(\text{ch}) = 0
\end{cases}
\]

Table 1: DLM-based feature templates
Results from Chen et al. 2012

- On PTB

<table>
<thead>
<tr>
<th>Order1</th>
<th>UAS</th>
<th>Order2</th>
<th>UAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>MST1</td>
<td>90.95</td>
<td>MST2</td>
<td>91.71</td>
</tr>
<tr>
<td>MST-DLM1</td>
<td>91.89</td>
<td>MST-DLM2</td>
<td>92.34</td>
</tr>
<tr>
<td>MSTB1</td>
<td>91.92</td>
<td>MSTB2</td>
<td>92.10</td>
</tr>
<tr>
<td>MSTB-DLM1</td>
<td>92.55</td>
<td>MSTB-DLM2</td>
<td>92.76</td>
</tr>
</tbody>
</table>

Table 5: Main results for English
- Partial tree level

- Other approaches
- Partial tree level

- Hybrid discriminative and generative models (Suzuki et al., 2009)
  - No explicit parses are produced for the unlabeled data.
  - Instead, derive useful estimations from the unlabeled data using many generative models, and integrate them in a discriminative model.
    - Many generative models
      - Each estimates the possibility of a given link from the view of one single feature (e.g., the word-bigram feature)
      - Trained with the unlabeled data.
    - A discriminative model (CRF)
      - Combine the basic features and the estimations of the generative models.
      - Trained with the labeled data.
- Suzuki et al., 2009

- Hybrid discriminative and generative models (Suzuki et al., 2009)

- Iteratively Training

  - Step 1: train the discriminative model with the labeled data adopting uniform distributions for the parameters of the generative models.

  - Step 2: apply the EM algorithm to train the generative models where the distributions are determined by the above discriminative model.

  - Step 3: retrain the discriminative model adopting the above generative models. Go to step 2 if desired.
Results from Suzuki et al., 2009

<table>
<thead>
<tr>
<th>dependency parser</th>
<th>test</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(McDonald et al., 2005a)</td>
<td>90.9</td>
<td>1od</td>
</tr>
<tr>
<td>(McDonald and Pereira, 2006)</td>
<td>91.5</td>
<td>2od</td>
</tr>
<tr>
<td>(Koo et al., 2008)</td>
<td>92.23</td>
<td>1od, 43M ULD</td>
</tr>
<tr>
<td>SS-SCM (w/ CL)</td>
<td><strong>92.70</strong></td>
<td>1od, 3.72G ULD</td>
</tr>
<tr>
<td>(Koo et al., 2008)</td>
<td>93.16</td>
<td>2od, 43M ULD</td>
</tr>
<tr>
<td>2-stage SS-SCM(+MIRA, w/ CL)</td>
<td><strong>93.79</strong></td>
<td>2od, 3.72G ULD</td>
</tr>
</tbody>
</table>
- Partial tree level

- Meta features (Chen et. al, 2013)
  - A base feature represents a kind of partial tree structure
  - The base features may suffer from the data sparseness problem
  - The target is to build connections among base features
Transformation function

- Key
  - Transformation function: $m_{f_b} = \Phi(f_b)$
Transformation Functions

- Options
  - PCA-based algorithms
  - Clustering algorithms
  - Rule-based approaches
  - ...

Feature clustering → OK
Transformation Function

- A simple mapping function (Chen et al. 2013)
  - Generate base features from the trees in the auto-parsed data
  - Collect the features and count their frequencies
  - Sort the collected features in decreasing order
  - $R(f_b)$ is the position number of $f_b$ in the list

\[
\Phi(f_b) = \begin{cases} 
H_i & \text{if } R(f_b) \leq \text{TOP10} \\
M_i & \text{if } \text{TOP10} < R(f_b) \leq \text{TOP30} \\
L_i & \text{if } \text{TOP30} < R(f_b) \\
O_i & \text{Others}
\end{cases}
\]
Meta-features

- Generating meta features

I ate the meat with a fork.

\( T_k : h_w, d_w, c_w, d(h,d,c) \)

\( F_b : \) ate, meat, with, RIGHTSIB

\( \Phi (f_b) = M_k \)

\([M_k]; [M_k], VV; [M_k], ate\)
Results from Chen et al., 2013

- On PTB

<table>
<thead>
<tr>
<th></th>
<th>UAS</th>
<th>COMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>92.76</td>
<td>48.05</td>
</tr>
<tr>
<td>MetaParser</td>
<td>93.77</td>
<td>51.36</td>
</tr>
</tbody>
</table>

Table 7: Main results on English
Semi-supervised dependency parsing

- Whole tree level
- Partial tree level
- Lexical level
Lexical level

• Learn information from words
  • Word clusters as extra features (Koo et al., 2008)

![Figure from (Koo et al., 2008)]

• Lexical co-occurrences counts from the web
  • PMI (Zhou et al., 2011)
  • Web count (Bansal and Klein, 2011)
Lexical level

- Learn information from words
  - Word clusters as extra features (Koo et al., 2008)

- Each word can be represented as a bit string
- By using prefixes of various lengths, we can produce different clusters

Figure from (Koo et al., 2008)
Word clusters

• Different types of word clusters
  • c4: 4 bit-string prefix
  • c6: 6 bit-string prefix
  • c*: full bit-string. 1,000 distinct bit-strings
Word clusters

- Examples
  - $ht, mt$: head POS + modifier POS
  - $hc4, mc4$: 4 bit prefix of head + 4 bit prefix of modifier

<table>
<thead>
<tr>
<th>Baseline</th>
<th>Cluster-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ht, mt$</td>
<td>$hc4, mc4$</td>
</tr>
<tr>
<td>$hw, mw$</td>
<td>$hc6, mc6$</td>
</tr>
<tr>
<td>$hw, ht, mt$</td>
<td>$hc*, mc*$</td>
</tr>
<tr>
<td>$hw, ht, mw$</td>
<td>$hc4, mt$</td>
</tr>
<tr>
<td>$ht, mw, mt$</td>
<td>$ht, mc4$</td>
</tr>
<tr>
<td>$hw, mw, mt$</td>
<td>$hc6, mt$</td>
</tr>
<tr>
<td>$hw, ht, mw, mt$</td>
<td>$ht, mc6$</td>
</tr>
<tr>
<td>$...$</td>
<td>$hc4, mw$</td>
</tr>
<tr>
<td></td>
<td>$hw, mc4$</td>
</tr>
<tr>
<td></td>
<td>$...$</td>
</tr>
<tr>
<td>$ht, mt, st$</td>
<td>$hc4, mc4, sc4$</td>
</tr>
<tr>
<td>$ht, mt, gt$</td>
<td>$hc6, mc6, sc6$</td>
</tr>
<tr>
<td>$...$</td>
<td>$ht, mc4, sc4$</td>
</tr>
<tr>
<td></td>
<td>$hc4, mc4, gc4$</td>
</tr>
<tr>
<td></td>
<td>$...$</td>
</tr>
</tbody>
</table>
Results from Koo et al. 2008

- On PTB

<table>
<thead>
<tr>
<th>Sec</th>
<th>dep1</th>
<th>dep1c (±)</th>
<th>dep2</th>
<th>dep2c (±)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>90.48</td>
<td>91.57 (+1.09)</td>
<td>91.76</td>
<td>92.77 (+1.01)</td>
</tr>
<tr>
<td>01</td>
<td>91.31</td>
<td>92.43 (+1.12)</td>
<td>92.46</td>
<td>93.34 (+0.88)</td>
</tr>
<tr>
<td>23</td>
<td>90.84</td>
<td>92.23 (+1.39)</td>
<td>92.02</td>
<td>93.16 (+1.14)</td>
</tr>
<tr>
<td>24</td>
<td>89.67</td>
<td>91.30 (+1.63)</td>
<td>90.92</td>
<td>91.85 (+0.93)</td>
</tr>
</tbody>
</table>
Lexical Level

- Word co-occurrences (Zhou et al. 2011)
  - Web-scale resources
    - N-gram counts by search engine google.
    - N-gram counts by google Web 1T 5-gram corpus
  - Association score between two words

\[ PMI(x, y) = \log \frac{p(\text{“}x\text{ “}y\text{ “})}{p(\text{“}x\text{ “})p(\text{“}y\text{ “})} \]
### Word co-occurrences

<table>
<thead>
<tr>
<th>N-gram feature templates</th>
</tr>
</thead>
<tbody>
<tr>
<td>hw, mw, PMI(hw,mw)</td>
</tr>
<tr>
<td>hw, ht, mw, PMI(hw,mw)</td>
</tr>
<tr>
<td>hw, mw, mt, PMI(hw,mw)</td>
</tr>
<tr>
<td>hw, ht, mw, mt, PMI(hw,mw)</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>hw, mw, sw</td>
</tr>
<tr>
<td>hw, mw, sw, PMI(hw, mw, sw)</td>
</tr>
<tr>
<td>hw, mw, gw</td>
</tr>
<tr>
<td>hw, mw, gw, PMI(hw, mw, gw)</td>
</tr>
</tbody>
</table>
### Results from Zhou et al. 2011

- On PTB

<table>
<thead>
<tr>
<th>Sec</th>
<th>dep1</th>
<th>+hits</th>
<th>+V1</th>
<th>dep2</th>
<th>+hits</th>
<th>+V1</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>90.39</td>
<td>90.94</td>
<td>90.91</td>
<td>91.56</td>
<td>92.16</td>
<td>92.16</td>
</tr>
<tr>
<td>01</td>
<td>91.01</td>
<td>91.60</td>
<td>91.60</td>
<td>92.27</td>
<td>92.89</td>
<td>92.86</td>
</tr>
<tr>
<td>23</td>
<td>90.82</td>
<td>91.46</td>
<td>91.39</td>
<td>91.98</td>
<td>92.64</td>
<td>92.59</td>
</tr>
<tr>
<td>24</td>
<td>89.53</td>
<td>90.15</td>
<td>90.13</td>
<td>90.81</td>
<td>91.44</td>
<td>91.41</td>
</tr>
</tbody>
</table>
Semi-supervised dependency parsing (In-domain)

- Three levels
  - Lexical level
  - Partial tree level
  - Whole tree level
End of Part B
References

References

References


Part C: Parsing the web and domain adaptation
In-domain vs out-domain

Annotated data in Domain A

Training

A Parser

Parsing texts in Domain A

Parsing texts in Domain B

In-domain

Out-domain
Motivation

- Few or no labeled resources exist for parsing text of the target domain.

- Unsupervised grammar induction?
  - Lots of work
  - Accuracies significantly lag behind those of supervised systems
  - Only on short sentences or assuming the existence of gold POS tags

- Build strong parsers by exploring labeled resources of existing domains plus unlabeled data for the target domain.
Outline

• Three shared tasks for parsing out-domain text
• Approaches for parsing out-domain text
  • news domain
  • web data
Shared tasks

- CoNLL 2007 shared task on domain adaptation
- CoNLL 2009 shared task on domain adaptation
- SANCL 2012 parsing the web
CoNLL 2007 shared task on domain adaptation

- Setup for the domain adaptation track
  - Data
    - Train: Large-scale labeled data for the source domain (WSJ)
    - Development: labeled data for biomedical abstracts
    - Test: labeled data for chemical abstracts
    - Unlabeled: large-scale unlabeled data for each train/dev/test.
  - The goal is to use the labeled data of the source domain, plus any unlabeled data, to produce accurate parsers for the target domains.
CoNLL 2009 shared task on domain adaptation

- Setup for the domain adaptation track
  - Czech, German, English (Brown corpus)
  - No unlabeled data
- Provide initial out-of-domain results for the three languages.
SANCL 2012: Parsing the web

- Data Setup (Petrov and McDonald, 2012)
  - Labeled data
    - Train: WSJ-train
    - Development: emails, weblogs, WSJ-dev
    - Test: answers, newsgroups, reviews, WSJ-test
  - Unlabeled data
    - Large-scale unlabeled data for all domains
- The goal is to build a single system that can robustly parse all domains.
## Data sets for SANCL 2012

<table>
<thead>
<tr>
<th>Training</th>
<th>Development</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSJ-train</td>
<td>Emails</td>
<td>Weblogs</td>
</tr>
<tr>
<td>Sentences</td>
<td>30,060</td>
<td>2,450</td>
</tr>
<tr>
<td>Tokens</td>
<td>731,678</td>
<td>29,131</td>
</tr>
<tr>
<td>Types</td>
<td>35,933</td>
<td>5,478</td>
</tr>
<tr>
<td>OOV</td>
<td>0.0%</td>
<td>30.7%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Emails</th>
<th>Weblogs</th>
<th>Answers</th>
<th>Newsgroups</th>
<th>Reviews</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentences</td>
<td>1,194,173</td>
<td>524,834</td>
<td>27,274</td>
<td>1,000,000</td>
<td>1,965,350</td>
</tr>
<tr>
<td>Tokens</td>
<td>17,047,731</td>
<td>10,356,284</td>
<td>424,299</td>
<td>18,424,657</td>
<td>29,289,169</td>
</tr>
<tr>
<td>Types</td>
<td>221,576</td>
<td>166,515</td>
<td>33,325</td>
<td>357,090</td>
<td>287,575</td>
</tr>
</tbody>
</table>
Approaches for parsing canonical out-domain text (CoNLL07)

- Feature-based approaches
  - Only include features that transfer well (Dredze+, 07)
  - Structural corresponding learning: transform features from source domain to target domain (Shimizu and Nakagawa, 07)

- Ensemble-based approaches
  - Stacking (Dredze+, 07)
  - Co-training (Sagae and Tsujii, 07)
  - Variant of self-training (Watson and Briscoe, 07)
Approaches for parsing canonical out-domain text (CoNLL07)

- Other approaches
  - Tree revision rules for target domain (Attardi+, 07)
  - Training instance weighting (Dredze+, 07)
  - Hybrid: use the output of a Constraint Grammar parser (Bick, 07)
  - Use collocations and relational nouns from unlabeled target domain data (Schneider+, 07)
Frustratingly hard domain adaptation 
(Dredze+, 2007)

- Theoretical work on domain adaptation attributes 
adaptation loss to two sources (Ben-David+, 2006)
  - Difference in the distribution between domains
  - Difference in labeling functions
- The error analysis of Dredze+ (2007) suggests that 
the primary source of errors is the difference in 
anotation guidelines between treebanks.
Frustratingly hard domain adaptation (Dredze+, 2007)

- Challenges for adaptation from WSJ (90%) to BIO (84%)
  - Annotation divergences between BIO and WSJ
  - Unlike WSJ, BIO contains many long sequence of digits.
  - Complex noun phrases
  - Appositives
  - WSJ uses fine-grained POS tags such as NNP, while BIO uses NN.
- Long list of failed attempts
Frustratingly hard domain adaptation (Dredze+, 2007)

- Feature manipulation
  - Remove features less likely to transfer
  - Add features more likely to transfer
  - Using word clustering features

- Parser diversity
  - Ensemble of parsers (similar to stacking and bagging)

- Target focused learning
  - Assign higher weights to instances similar to the target when training
Domain + non-canonical text differences

Domain A
Canonical text

Domain B

Domain B
Non-canonical text
Parsing non-canonical out-domain text (SANCL)

- What is new?
  - Inconsistent usage of punctuation and capitalization
  - Lexical shift due to increased use of slang, technical jargon, or other phenomena.
  - Spelling mistakes and ungrammatical sentences
  - Some syntactic structures are more frequently used in web texts than in newswire
    - Questions, imperatives, long lists of names, sentence fragments…
Examples

- Plz go there.
- I like it very much!!!!!!
- Gooooooood
- ...

...
Approaches for parsing non-canonical out-domain text (SANCL)
Approaches

Domain B
Non-canonical

text

Domain B
Canonical text

Domain A
Canonical text

Text normalization

Domain adaptation
Approaches for parsing non-canonical out-domain text (SANCL)

- Main approaches
  - Text normalization (preprocessing)
  - Ensemble of parsers
  - Self-training for constituent parsing
  - Word clustering/embedding
  - Co/tri-training (unsuccessful)
  - Instance weighting and genre classification
Text normalization

- Preprocessing the data leads to better POS tagging and parsing performance. (Foster, 2010; Gadde+, 2011; Roux and Foster+, 2012)
Text normalization

- The preprocessing rules of (Roux, Foster+, 2012)
  - Emoticon => comma or full stop
  - Email address, URL => generic strings
  - Uppercased words => lowercased
  - Abbreviations, spelling variants (plz, ppl) => standard form
  - nt; s => n’t; ’s
  - Repeated punctuation (!!!) => collapsed into one
  - List items (# 2) => removed
Text normalization

- The preprocessing rules of (Seddah+, 2012)
  - An Ontonote/PTB token normalization stage
  - Smileys, URLs, email addresses, similar entities
  - Correct tokens or token sequences
    - Spelling error patterns
    - Lowercasing
    - Rewriting rules for dealing with frequent amalgams (gonna or im)

<table>
<thead>
<tr>
<th></th>
<th>Ontonotes dev all</th>
<th>e-mail dev all</th>
<th>weblog dev all</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>unk</td>
<td>unk</td>
<td>unk</td>
</tr>
<tr>
<td>-corr</td>
<td>96.5</td>
<td>92.3</td>
<td>94.7</td>
</tr>
<tr>
<td>+corr</td>
<td>96.5</td>
<td>92.9</td>
<td>94.7</td>
</tr>
</tbody>
</table>
Text normalization

- The preprocessing rules of (McClosky+, 2012)
  - High-precision text replacements
    - 1,057 spelling auto-correction rules (yuo => you) from Pidgin instant messaging client
    - 151 common Internet abbreviations (LOL => “laughing out loud”)
  - Limited gain
    - Such spelling errors are infrequent in the unlabeled data.
Ensemble of parsers

- Product-of-experts (Alpage, DCU-Paris13)
- Stacking (IMS, Stanford, UPenn)
- Voting (CPH-Trento, DCU-Paris, HIT)
- Bagging (HIT)
- Up-training (IMS)
- Re-ranking (DUC-Paris13, IMS, Stanford)
- Model merging (OHSU, Stanford)

- Obtain large improvement gain.
  - More like improvement in in-domain parsing
  - Contribution to domain adaptation?
Exploring unlabeled data

- Self-training (successful for constituent parsers)
  - Two-stage generative model and reranker (Charniak and Johnson, 2005)
  - Generative PCFG-LA model (Petrov and Klein, 2007)
- Word clusters or embeddings
- Co/tri-training (unsuccessful for dependency parsers)
Why self-training is unsuccessful for dependency parsing?

- Generative models suffer less from the over-fitting problem during training.
- Current dependency parsing models are commonly discriminative.
  - Linear models with online training, no probabilistic explanation.
  - Generative models leads to unsatisfactory accuracy.
### Evaluation results

- **Top 4 systems of SANCL on POS tagging**
- **Tagging performance is very important**

<table>
<thead>
<tr>
<th>Team</th>
<th>Answers</th>
<th>Newsgroups</th>
<th>Reviews</th>
<th>WSJ</th>
<th>Averaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCU-Paris (Roux, Foster+)</td>
<td>91.79</td>
<td>93.81</td>
<td>93.11</td>
<td>97.29</td>
<td>92.90 (1)</td>
</tr>
<tr>
<td>HIT (Zhang+)</td>
<td>90.99</td>
<td>93.32</td>
<td>90.65</td>
<td>97.76</td>
<td>91.32 (2)</td>
</tr>
<tr>
<td>IMS (Bohnet+)</td>
<td>91.07</td>
<td>91.70</td>
<td>90.01</td>
<td>97.57</td>
<td>90.93 (3)</td>
</tr>
<tr>
<td>Stanford (McClosky+)</td>
<td>90.30</td>
<td>91.49</td>
<td>90.46</td>
<td>95.00</td>
<td>90.75 (4)</td>
</tr>
</tbody>
</table>
Which one is the best/most important?

- Main approaches
  - Text normalization (preprocessing)
  - Ensemble of parsers
  - Self-training for constituent parsing
  - Word clustering/embedding
  - Co/tri-training (unsuccessful)
  - Instance weighting and genre classification
End of Part C
References


References

References


References

- Djame Seddah, Benoit Sagot, and Marie Candito. 2012. Robust pre-processing and semi-supervised lexical bridging for user-generated content parsing. In Notes of the First Workshop on SANCL.
References

- Meishan Zhang, Wanxiang Che, Yijia Liu, Zhenghua Li, Ting Liu. 2012. HIT dependency parsing: Bootstrap aggregating heterogeneous parsers. In Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)
Part D: multilingual dependency parsing
Motivation

• A difficult syntactic ambiguity in one language may be easy to resolve in another language (bilingual constraints)
• A more accurate parser on one language may help a less accurate one on another language (semi-supervised)
• Rich labeled resources in one language can be transferred to build parsers of another language (unsupervised)
Obstacles

- Syntactic non-isomorphism across languages
- Different annotation choices (guideline)
- Partial (incomplete) parse trees resulted from projection
- Parsing errors on the source side
- Word alignment errors
Research map (two perspectives)

• Methods
  • Delexicalized (not rely on bitext)
  • Projection (rely on bitext)
    • Project hard edges
    • Project edge weights (distributions)
    • As training objective (bilingual constraints)

• Supervision (existence of target treebank)
  • Semi-supervised
  • Unsupervised
Bilingual word reordering features (Huang+ 09)

- Supervised with bilingual constraints
- Use the word reordering information in source language as extra features when parsing target text (using automatic word alignment as bridge)

Two characteristics
- Bilingual text with target-side labels
- No parse tree in the source side
Build local classifiers via projection (Jiang & Liu 10)

- Semi-supervised; project edges
  - Step 1: projection to obtain dependency/non-dependency classification instances
  - Step 2: build a target-language local dependency/non-dependency classifier
  - Step 3: feed the outputs of the classifier into a supervised parser as extra weights during test phase.
Prune projected structures with target-language marginal (Li+ 14)

- Semi-supervised; project edges from EN to CH
- The goal is to acquire extra large-scale high-quality labeled training instances from bitext

- Step 1: parse English sentences of bitext
- Step 2: project English trees into Chinese, and filter unlikely dependencies with target-language marginal probabilities (handling cross-language non-isomorphism, or parsing and alignment errors)
- Step 3: train with extra training instances with partial annotations.
Prune projected structures with target-language marginal (Li+ 14)

(a) Source tree and word alignments

(b) Projected incomplete tree

(c) Forest (ambiguous labelings)
Delexicalized transfer (no bitext) (McDonald+ 11)

- Unsupervised; delexicalized
- Direct delexicalized transfer (Zeman & Resnik 08, Sogaard 11, Cohen+ 11)
  - Train a delexicalized parser on source treebank, ignoring words, only using universal POS tags (Petrov+ 11)
Delexicalized transfer (no bitext) (McDonald+ 11)

- **Direct** delexicalized transfer
  - Step 1: replace the language-specific POS tags in source treebank with the universal POS tags (Petrov+ 11)
    - NOUN (nouns); VERB (verbs); ADJ (adjectives); ADV (adverbs)
    - PRON (pronouns); DET (determiners); ADP (pre/postpositions); NUM (numerals); CONJ (conjunctions); PRT (particles); PUNC (punctuation marks); X (a catch-all tag)
  - Step 2: train a delexicalized source-language parser
    - Only use POS tag features (no lexical features)
    - 89.3 (lex) => 82.5 (delex) on English test set
  - Step 3: parse target-language sentences with universal POS tags
Delexicalized transfer (no bitext) (McDonald+ 11)

- **Multi-source** delexicalized transfer
  - Use multiple source treebanks (besides English), and concatenate them as a large training data.
  
- Can improve parsing accuracy.

![Diagram of delexicalized data with English, Greek, German, and other languages, leading to a parser for Danish](image)
Delexicalized + projected transfer (with bitext) (McDonald+ 11)

- Unsupervised; delexicalized + project (as training objective)

- Workflow (Danish as target language)
  - Step 1: parse a set of Danish sentences with the delexicalized transfer parser (trained on En treebank).
  - Step 2: train a lexicalized Danish parser using the predicted parses as gold-standard.
  - Step 3: improve the lexicalized Danish parser on bitext
    - Extrinsic objective: the parser’s outputs are more consistent with the outputs of the lexicalized English parser (training with multiple objectives, Hall+ 11).
Delexicalized transfer with cross-lingual word clusters (Tackstrom+ 12)

- Unsupervised; delexicalized + cross-lingual word clusters

- In delexicalized dependency parsers
  - Only POS tag features are used

- In other previous work on dependency parsing
  - Features based on word clusters work very well
Delexicalized transfer with cross-lingual word clusters (Tackstrom+ 12)

- Word clusters derived from large-scale unlabeled data can alleviate data sparseness.
- POS tags < word clusters < words

Accuracy (LAS) of supervised English Parser
Delexicalized transfer with cross-lingual word clusters (Tackstrom+, 2012)

- Re-lexicalize the delexicalized transfer parser by using cross-lingual word clusters.
- Cross-lingual word clusters need to be consistent across languages.
- A cross-lingual clustering algorithm that leverages
  - large amounts of monolingual unlabeled data, and
  - parallel data through which the cross-lingual word cluster constrains are enforced.
Delexicalized transfer with cross-lingual word clusters (Tackstrom+ 12)

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Lang.</th>
<th>Sample words</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>EN</td>
<td>was, wasn’t, wasn’t,</td>
</tr>
<tr>
<td>60</td>
<td>ES</td>
<td>estaba, estarán, est,</td>
</tr>
<tr>
<td>101</td>
<td>EN</td>
<td>very, mildly, wholly,</td>
</tr>
<tr>
<td>101</td>
<td>ES</td>
<td>muchomás, fuerte, f</td>
</tr>
<tr>
<td>153</td>
<td>EN</td>
<td>chicken, bird, ostric</td>
</tr>
<tr>
<td>153</td>
<td>ES</td>
<td>pollo, achote, manz:</td>
</tr>
<tr>
<td>195</td>
<td>EN</td>
<td>The, ...</td>
</tr>
<tr>
<td>195</td>
<td>ES</td>
<td>El, La, Los, Las, Lo</td>
</tr>
<tr>
<td>236</td>
<td>EN</td>
<td>dry, wet, moist, lifel</td>
</tr>
<tr>
<td>236</td>
<td>ES</td>
<td>seco, secos, semiseco</td>
</tr>
</tbody>
</table>
Delexicalized Treebank

Cluster  Lang.  Sample words
60       EN     was, wasn’t, wasn’t, wasn’t, hasn’t, doesn’t, ...
60       ES     estaba, estarán, estuvo, fui, quedaba, ...

Delexicalized Parser

Add tags & clusters

El    pollo    estaba    muy    seco

Train

Delexicalized Parser

DT      NOUN    VERB    ADV    ADJ
195     153     60      101    236

Parse

El    pollo    estaba    muy    seco

DT      NOUN    VERB    ADV    ADJ
195     153     60      101    236
Selectively multi-source delexicalized transfer (Tackstrom+ 13)

- Unsupervised; delexicalized (also in Naseem+ 12)
- Some delexicalized features do not transfer well across typologically different languages
- Selectively parameter sharing based on typological and language-family features (Dryer and Haspelmath 11)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>81A</td>
<td>Order of Subject, Object and Verb</td>
</tr>
<tr>
<td>85A</td>
<td>Order of Adposition and Noun</td>
</tr>
<tr>
<td>86A</td>
<td>Order of Genitive and Noun</td>
</tr>
<tr>
<td>87A</td>
<td>Order of Adjective and Noun</td>
</tr>
</tbody>
</table>
Selectively multi-source delexicalized transfer (Tackstrom+ 13)

- Selectively use features for different source and target language pairs
  - Indo-European (Bulgarian, Catalan, Czech, Greek, English, Spanish, Italian, Dutch, Portuguese, Swedish)
  - Altaic languages (Japanese, Turkish)
Ambiguity-aware transfer (Tackstrom+ 13)

- Unsupervised; delexicalized

- Relexicalize delexicalized parsers using target-language unlabeled text.
  - First, parse the unlabeled data with the base delexicalized parser, and use the arc-marginals to construct ambiguous labels (forest) for each sentence.
  - Then, train a lexicalized parser on the unlabeled data via ambiguity-aware training.
Ambiguity-aware transfer (Tackstrom+ 13)

- Ambiguity-aware training
  - For each training instance \( x \), a set of ambiguous labels \( \tilde{\mathbf{y}}(x) \) are provided, instead of a gold-standard label.

\[
\mathcal{L}(\theta; \tilde{\mathcal{D}}) = \sum_{i=1}^{n} \log \left\{ \sum_{y \in \tilde{\mathbf{y}}(x_i)} p(\theta(y | x_i)) \right\} - \lambda \|\theta\|_2^2
\]
Ambiguity-aware transfer (Tackstrom+ 13)

- Ambiguity-aware self-training
- Ambiguity-aware ensemble training
  - Add the outputs of another parser (Naseem+, 2012) into the ambiguous label set.
  - Can improve accuracy.
Syntax projection as posterior regularization (Ganchev+ 09)

- Unsupervised; project high-probability edges (used in training objective)
- Key points
  - Filter unlikely projected dependencies according to source-side dependency probabilities and alignment probabilities (only project high-confidence dependencies)
  - Give the model more flexibility via posterior regularization
Syntax projection as posterior regularization (Ganchev+ 09)

- Posterior regularization
  - After training, the model should satisfy that the expectation of the number of conserved edges (among the projected edges) is at least 90%
Syntax projection as posterior regularization (Ganchev+ 09)

\[ \mathbb{E}[-\log p_\theta(x)] + R(\theta) + \mathbb{E}[\text{KL}(Q_x \| p_\theta(z \mid x))] \]

- log-likelihood of the target-side bitext
- Model prior
- The set of distributions satisfying the desired posterior constrains
- Model posterior
Syntax projection as posterior regularization (Ganchev+ 09)

- Create several simple rules to handle different annotation decisions (annotation guideline).
  - E.g., Main verbs should dominate auxiliary verbs.

- Can largely improve parsing accuracy.
Combining unsupervised and projection objectives (Liu+ 13)

- Unsupervised; project edges
- Derive local dependency/non-dependency instances from projection (Jiang+, 10)
Combining unsupervised and projection objectives (Liu+ 13)

- Workflow
Combining unsupervised and projection objectives (Liu+ 13)

• Joint objective: linear interpolation of

Unsupervised objective

$$\theta(\lambda) = \prod_{d_e \in D_E} Pr(+) | d_e \prod_{d_e \in \tilde{D}_E} Pr(- | d_e)$$

Projection objective

$$\phi(\lambda) = \sum_{d_e \in D_P} \log Pr(+) | d_e + \sum_{d_e \in D_N} \log Pr(- | d_e)$$
Transfer distribution as bilingual guidance (Ma & Xia 14)

- Unsupervised; hybrid of projection and delexicalized; transfer edge weights (distribution)

\[
\tilde{w}(e^t, x_i^t) = \begin{cases} 
  w_E(e^s, x_i^s), & \text{if } e^t \xrightarrow{align} e^s \\
  w_E(e^t_{delex}, x_i^s), & \text{otherwise}
\end{cases}
\]

(a) Source tree and word alignments

(b) Projected incomplete tree
Transfer distribution as bilingual guidance (Ma & Xia 14)

- Use the target-language sentences to train a parser, by minimizing the KL divergence from the transferred distribution to the distribution of the learned model.

\[
\sum_{y_i} \tilde{p}(y_i | x_i) \log p_{\lambda}(y_i | x_i)
\]
Transfer distribution as bilingual guidance (Ma & Xia 14)

- Use unlabeled data by minimizing entropy.
- Not very helpful
Summary of Part D

- Help target-language parsing with source-language treebanks and bitext
  - Semi-supervised vs. unsupervised
  - Projection vs. delexicalized
  - Project hard edges, transfer edge weights, or use bilingual constraints
Summary of Part D

- Future thoughts
  - Word alignment errors (probabilities)
  - Source-language parsing errors (probabilities)
  - How to capture cross-language non-isomorphism
  - Joint word alignment and bilingual parsing?
References

- Xuezhe Ma and Fei Xia. 2014. Unsupervised Dependency Parsing with Transferring Distribution via Parallel Guidance and Entropy Regularization. In ACL
References

Part E: conclusion and discussions
Topics in this talk

- Dependency parsing and supervised approaches
  - Single model
    - Graph-based; Transition-based; Easy-first; Constituent-based
  - Hybrid model
  - Non-projective dependency parsing
Topics in this talk

- Semi-supervised approaches for in-domain text
  - Whole tree level
  - Partial tree level
  - Lexical level
Topics in this talk

- Approaches for parsing out-domain text
  - Shared tasks for parsing domain adaptation
  - Parsing canonical out-domain text
  - Parsing non-canonical out-domain text (web data)
    - Text normalization is important
- Multilingual dependency parsing
Discussions

- Supervised track
  - Faster decoding algorithm for higher-order graph-based models
  - Broader search space for transition-based models
  - Theoretical and empirical comparison of dependency and phrase-structure parsers
    - Results indicate that the phrase-structure parsers can produce better syntactic structures than dependency parsers.
Discussions

- Semi-supervised track
  - The approaches of Lexical or partial tree levels work well in exploring unannotated data
  - The approaches of whole-tree level are not effective
    - Exception: ambiguity-aware ensemble training
  - More effective semi-supervised approaches
    - How to select reliable sentences/fragments?
    - Sentence > fragment > subtree > word
Discussions

• Parser domain adaptation
  • How to capture the domain differences, and then improve the models for the target domain?
  • How to find and extract features from unlabeled data which are helpful for target domain parsing?

• Parsing the web
  • Text normalization resources and procedures
Discussions

- Multilingual dependency parsing
  - Word alignment errors (probabilities)
  - Source-language parsing errors (probabilities)
  - How to capture cross-language non-isomorphism
  - Joint word alignment and bilingual parsing?
The End

- Thanks a lot.

- Q & A