Unsupervised Tree Induction for Tree-based Translation

Feifei Zhai
2013-7-3
Outline

- Introduction
- Tree induction via EM algorithm
- Tree induction via Bayesian inference
- Experiments
- Conclusion
Outline

- Introduction
- Tree induction via EM algorithm
- Tree induction via Bayesian inference
- Experiments
- Conclusion
Rule Extraction Process of Current Tree-based Translation (string-to-tree model)

- Parse target sentences by a supervised parser
- Achieve automatic word alignment
- Rule extraction
Challenges on Parse Trees

- Need Tree-Bank resources for training parsers
Challenges on Parse Trees

- Need Tree-Bank resources for training parsers
- Word alignment is separated with parsing process
Using Unsupervised Trees to Substitute Parse Trees

- Target sentence
- Unsupervised tree
- Source sentence
- Word alignment
- Rule extraction
Using Unsupervised Trees to Substitute Parse Trees

- Beneficial for the translation between resource-poor languages
- High compatibility between word alignment and tree structures
Outline

- Introduction
- Tree induction via EM algorithm
- Tree induction via Bayesian inference
- Experiments
- Conclusion
How to Produce Unsupervised Trees?

- Generate a packed forest for each sentence pair
  - Encode all reasonable tree structures into the packed forest

- Adopt an EM algorithm to learn an effective synchronous tree substitution grammar (STSG) and then acquire Viterbi tree structures accordingly
Packed Forest Generation

How to create a packed forest for a sentence without any syntactic knowledge?

- How to label the forest nodes?
- How to group the words in the sentence to forest nodes?
How to label forest nodes?

- Using word classes of boundary words
  - An effective indicator for reordering (Xiong et al., 2006)
  - Successfully used in labeling hierarchical rules (Zollmann and Vogel, 2011)
Node Labeling Strategy

- Utilize POS tags as the word classes
- Divide the forest nodes into three groups:
  - one-word node
  - two-word node
  - multi-word node
Node Labeling Strategy

- Utilize POS tags as the word classes
- Divide the forest nodes into three groups:
  - one-word node
  - two-word node
  - multi-word node
Node Labeling Strategy

- Utilize POS tags as the word classes
- Divide the forest nodes into three groups:
  - one-word node
  - two-word node
  - multi-word node

Two-word node

```
we

meet

again
```
Node Labeling Strategy

- Utilize POS tags as the word classes
- Divide the forest nodes into three groups:
  - one-word node
  - two-word node
  - multi-word node

```
we
PRP

meet
VBP

again
RB
```

Multi-word node
How to group words?

- Assign each span a forest node?
  - $0.5L(L + 1)$ forest nodes
  - $\frac{1}{6}(L^3 + 5L)$ binary hyperedges
How to group words?

- Assign each span a forest node?
  - $0.5L(L+1)$ forest nodes
  - $\frac{1}{6}(L^3 + 5L)$ binary hyperedges

Too many STSG parameters!!
How to group words?

- Assign each span a forest node?
  - $0.5L(L+1)$ forest nodes
  - $\frac{1}{6}(L^3 + 5L)$ binary hyperedges

- Space reduction
  - Bilingual sentence segmentation
  - Frontier node assumption

Too many STSG parameters!!
Bilingual Sentence Segmentation

- Segment a sentence pair into several short sub-sentence pairs.
- A strategy based on punctuations and word alignment.

Today we meet again, but the situation is quite different.

今天 我们, 再次 见面, 情形 已 大 不 相同 了.
Today we meet again, but the situation is quite different.
Bilingual Sentence Segmentation

Today we meet again, but the situation is quite different.

今天 我们 再次 见面， 情形 已 大 不 相同 了
Bilingual Sentence Segmentation

Today we meet again, but the situation is quite different, zai-ci jian-mian qing-xing yi da bu xiang-tong le.
After bilingual sentence segmentation, we realign the words based on the sub-sentence pairs to get a new word alignment.
The more frontier nodes the tree structure has, the more reasonable it is for translation.
The more frontier nodes the tree structure has, the more reasonable it is for translation.
Frontier Node Assumption

- The more frontier nodes the tree structure has, the more reasonable it is for translation.
Frontier Node Assumption

- The more frontier nodes the tree structure has, the more reasonable it is for translation.

More frontier nodes are conducive to extracting many small minimal rules and generating larger rules by composition.
Frontier Node Assumption

The more frontier nodes the tree structure has, the more reasonable it is for translation.

Only consider the tree structures with the largest number of frontier nodes during forest construction.
Forest Constructor

- We take the binary structure as the basic unit of the packed forest.
  - It is very effective to improve the translation quality (Wang et al., 2007; Zhang et al., 2011)
  - It would be much easier to construct the forest if we only consider the binary structures.
Forest Constructor

- Bilingual sentence segmentation
- Perform a CKY-style algorithm to construct a sub-forest for each sub-sentence pair under the frontier node assumption
- Combine the sub-forests to generate the final forest for the whole sentence pair
Forest Constructor (CKY style algorithm)

- Inspects each span in a bottom-up manner and creates forest nodes to represent the spans.

Graph:

- Node 0: \(NN [0,1]\)
  - "Today" (jin-tian)
    - "今天"

- Node 1: \(PRP [1,2]\)
  - "we" (wo-men)

- Node 2: \(VBP [2,3]\)
  - "meet"
    - "zai-ci"
    - "再次"

- Node 3: \(VBP + RB [2,4]\)
  - "meet"
    - "again"
    - "jian-mian"
  - "见面"

- Node 4: \(RB [2,4]\)
  - "again"
    - "jian-mian"
    - "见面"
Forest Constructor (CKY style algorithm)

- For each span, check every split point of it and generate an binary edge for that split point.
- Span \([1,4]\) = \([1,2]\) + \([2,4]\)
Forest Constructor (CKY style algorithm)

- For each span, check every split point of it and generate an binary edge for that split point.
- Span $[1,4] = [1,3] + [3,4]$
Forest Constructor (CKY style algorithm)

- Only retain the edges that carries the largest number of frontier nodes.
Final Forest of the Sub-sentence Pair

Today we meet again.

今天 我们 再次 见面
Today we meet again, but the situation is quite different.
EM Algorithm

- Choose a series of trees from the packed forests by maximizing the likelihood of the whole corpus:

\[ (t_{e_1} t_{e_2} \ldots t_{e_n})^* = \arg \max_{(t_{e_1} t_{e_2} \ldots t_{e_n})} \prod_{i=1}^{n} p(t_{ei}, f_i, a_i) \]

- \( p(t_{ei}, f_i, a_i) \) is computed by aggregating the rule probabilities in each STSG derivation \( d \) in the set of all derivations \( D \).

\[ p(t_{ei}, f_i, a_i) = \sum_{D} \prod_{r \in d} p(r) \]
Implementation

- Minimal translation rules
- Synchronous derivation forest
- Inside-outside algorithm
Outline

■ Introduction
■ Tree induction via EM algorithm
■ Tree induction via Bayesian inference
■ Experiments
■ Conclusion
Drawbacks of EM

- Need to represent all derivations in a forest
- Too many heuristics
  - bilingual sentence splitting
  - frontier node assumption
- Cannot work on the full space
- Cannot use prior information of rules (rigid)
  - frontier node assumption is rigid
Bayesian Inference

- Have simple and efficient training algorithm
  - Gibbs sampling

- Easy to integrate the prior information of rules
  - DP prior

- Could scale the entire space of tree structures
The Bayesian Model

- Based on STSG generation process

\[ p(d) = \prod_{i=1}^{n} p(\text{rule}_i \mid c_i) \]

- Distributions over STSG rules, given the root non-terminal of tree fragment

\[ \text{rule} \mid c \sim G_c \]

\[ G_c \mid \alpha_c, P_0 \sim DP(\alpha_c, P_0(\cdot \mid c)) \]
The Bayesian Model

\[ \text{rule} \mid c \sim G_c \]
\[ G_c \mid \alpha_c, P_0 \sim DP(\alpha_c, P_0(c)) \]

Integrate over all possible values of \( G_c \)

\[
p(\text{rule}_i \mid r^{-i}, c, \alpha_c, P_0) = \frac{n_{\text{rule}_i}^{-i} + \alpha_c P_0(\text{rule}_i \mid c)}{n_c^{-i} + \alpha_c}
\]
The Bayesian Model

\[ p(\text{rule}_i \mid r^{-i}, c, \alpha_c, P_0) = \frac{n_{\text{rule}_i}^{-i} + \alpha_c P_0(\text{rule}_i \mid c)}{n_{\text{c}}^{-i} + \alpha_c} \]

- \( P_0(\cdot \mid c) \), base distribution, assigns a prior probability to an infinite number of rules
The Bayesian Model

\[ p(\text{rule}_i \mid r^{-i}, c, \alpha_c, P_0) = \frac{n^{-i}_{\text{rule}_i} + \alpha_c P_0(\text{rule}_i \mid c)}{n^{-i}_c + \alpha_c} \]

- \( P_0(\cdot \mid c) \), **base distribution**, assigns a prior probability to an infinite number of rules

- \( \alpha \), **concentration parameter**, controls the tendency of the model:
  - reusing existing rules
  - creating novel ones
Each rule consists of a tree fragment $frag$ and a source string $str$ in the model, thus we factor the prior probability as follows:

$$P_0(rule \mid c) = P(frag \mid c) P(str \mid frag)$$
Base Distribution $P_0$

- Good translation grammar
  - Small rules: for good generality
  - Large rules: for enough context information

$$P_0(\text{rule} \mid c) = P(\text{frag} \mid c) P(\text{str} \mid \text{frag})$$
Base Distribution $P_0$

- Good translation grammar
  - Small rules: for good generality
  - Large rules: for enough context information

$$P_0(\text{rule} \mid c) = P(\text{frag} \mid c) \cdot P(\text{str} \mid \text{frag})$$
Base Distribution $P_0$

$$P_0(rule | c) = P(frag | c) P(str | frag)$$
Base Distribution $P_0$

$$P_0(\text{rule} \mid c) = P(\text{frag} \mid c) \cdot P(\text{str} \mid \text{frag})$$
Utilize binary structures

Recursively check each non-terminal node, and decide whether to expand it or not.
1. What type is the node, one-word, two-word or multi-word non-terminal?
2. What tag is used to label the node?
3. Will it be expanded?

1. It is a one-word node
2. The used tag is RB
3. Not Expand this RB node
1. What type is the node, one-word, two-word or multi-word non-terminal?
2. What tag is used to label the node?
3. Will it be expanded?

1. It is a **multi-word node**
2. The used tag is **NN...VBP**
3. **Expand** this **NN...VBP** node
1. It is a two-word node
2. The used tag is $NN+PRP$
3. Not Expand this $NN+PRP$ node
1. What type is the node, one-word, two-word or multi-word non-terminal?
2. What tag is used to label the node?
3. Will it be expanded?

1. It is a one-word node
2. The used tag is **VBP**
3. **Not Expand** this **VBP** node
Base Distribution $P_0$

- What type is the node, one-word, two-word or multi-word non-terminal?
  - uniform distribution

- What tag is used to label the node?
  - uniform distribution

- Will it be expanded?
  - bernoulli distribution
Base Distribution $P_0$

- What type is the node, one-word, two-word or multi-word non-terminal?
  - uniform distribution

- What tag is used to label the node?
  - uniform distribution

- Will it be expanded?
  - bernoulli distribution

- The terminals are generated from an uniform distribution
Base Distribution $P_0$

- What type is the node, one-word, two-word or multi-word non-terminal?
  - *uniform distribution*

- What tag is used to label the node?
  - *uniform distribution*

- Will it be expanded?
  - *bernoulli distribution*

- The terminals are generated from an *uniform distribution*
Base Distribution $P_0$

$$P_0(\text{rule} \mid c) = P(\text{frag} \mid c) \, P(\text{str} \mid \text{frag})$$
Base Distribution $P_0$

$$P_0(rule \mid c) = P(frag \mid c) \cdot P(str \mid frag)$$

```
NN...RB
  /  \\
NN+PRP:x0 VBP:x1 RB:x3
  / \
  x0  x2  x1
```
Base Distribution $P_0$

$$P_0(\text{rule} \mid c) = P(\text{frag} \mid c) P(\text{str} \mid \text{frag})$$

- How many terminals will be generated?
- Generate the terminals
- Insert non-terminals into the $\text{str}$
Base Distribution $P_0$

- Each terminal is generated from an *uniform distribution*
- The number of terminals is drawn from a *poisson distribution*
- The insertion of non-terminal is drawn from an *uniform distribution* based on possible insert positions
Base Distribution $P_0$

- Each terminal is generated from an *uniform distribution*
- The number of terminals is drawn from a *poisson distribution*
- The insertion of non-terminal is drawn from an *uniform distribution* based on possible insert positions

*short string*
Training

■ A collapsed gibbs sampler
  - exchangeability of the model
  - scale the entire tree space of a sentence
A collapsed gibbs sampler
- exchangeability of the model
- scale the entire tree space of a sentence

Sampling operators
- Rotate operator
- Two-level-left-Rotate operator
- Two-level-right-Rotate operator
Rotate Operator

- **S-node**: the non-root node covering at least two words.
- **Three descendants of s-node (DC)**: its left child, right child and its sibling.
**Rotate Operator**

- **S-node**: the non-root node covering at least two words.
- **Three descendants of s-node (DC)**: its left child, right child and its sibling.

```
PRP+VBP
   /
  / \\
PRP we
   / \ \
  /   \\
VBP meet
   /     \\
   /       \\
RB again
```
Rotation Operator

- Each s-node has two states:
  - **Left state**: governs its left two DCs
  - **Right state**: governs its right two DCs
Rotation Operator

- Each s-node has two states:
  - **Left state**: governs its left two DCs
  - **Right state**: governs its right two DCs
**Rotation Operator – Left State**

We meet again.

**s-node**

- **we**
- **zai-ci**
- **jian-mian**

**Left State**

**r0:**

- **PRP:** x0
- **VBP:** x1
- **RB:** x2
we: PRP
to meet: VBP
again: RB

we: PRP
zai-ci: VBP
jian-mian: RB

旋转算子 - 右状态

PRP...RB

VBP+RB

s-node

Right State

r1:

PRP:x0
VBP+RB:x1

r2:

VBP:x0
RB:x1

we: PRP
zai-ci: VBP
jian-mian: RB

再次 见面: RB
Choose a sample from the up two states:

\[
p(\text{left state}) = \frac{p(r_0 \mid r^-)}{p(r_0 \mid r^-) + p(r_1, r_2 \mid r^-)}
\]

\[
p(\text{right state}) = \frac{p(r_1, r_2 \mid r^-)}{p(r_0 \mid r^-) + p(r_1, r_2 \mid r^-)}
\]

Draw a random number from a 0-1 uniform distribution and decide which sample to choose.
Two-level-Rotate Operator

- Take a pair of s-nodes in a parent-child relationship as a unit for sampling
  - Two-level-left-Rotate operator
  - Two-level-right-Rotate operator
Two-level-left-Rotate Operator

Left State

Today we meet again

Right State

Today we meet again
Two-level-right-Rotate Operator

Left State

Today
we
meet
gain

Right State

Today
we
meet
gain

PRP+VBP
PRP+RB
NN...RB
NN...RB

Outline

- Introduction
- Tree induction via EM algorithm
- Tree induction via Bayesian inference
- Experiments
- Conclusion
Experimental Setup

- Chinese-to-English translation
- Training set: FBIS corpus (~230K parallel sentence pairs)
- Language model: 5-gram trained on Xinhua portion of the GIGA WORD corpus and the target part of the training corpus
- Development set: NIST03
- Test sets: NIST04, NIST05
Experimental Setup

- Translation System
  - Joshua
  - Moses-chart
  - s2t (parse trees)
  - s2t (B-parse tree, i.e., binarized parse trees)
  - s2t (EM-tree, i.e., unsupervised trees from EM algorithm)
  - s2t (Bayes-tree, i.e., unsupervised trees from Bayesian inference)
### Experimental Results

<table>
<thead>
<tr>
<th>System</th>
<th>MT04</th>
<th>MT05</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joshua</td>
<td>30.71</td>
<td>27.86</td>
<td>29.59</td>
</tr>
<tr>
<td>Moses-chart</td>
<td>32.63</td>
<td>30.05</td>
<td>31.66</td>
</tr>
<tr>
<td>s2t (parse tree)</td>
<td>33.73</td>
<td>30.25</td>
<td>32.75</td>
</tr>
<tr>
<td>s2t (B-parse tree)</td>
<td>34.09</td>
<td>30.99</td>
<td>32.92</td>
</tr>
<tr>
<td>s2t (EM-tree, re-aligned)</td>
<td>34.71</td>
<td>31.55</td>
<td>33.53</td>
</tr>
<tr>
<td>s2t (Bayes-tree)</td>
<td>35.20</td>
<td>31.69</td>
<td>33.89</td>
</tr>
<tr>
<td>Number of sentences</td>
<td>1788</td>
<td>1082</td>
<td>2870</td>
</tr>
</tbody>
</table>
### Experimental Results

<table>
<thead>
<tr>
<th>System</th>
<th>MT04</th>
<th>MT05</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joshua</td>
<td>30.71</td>
<td>27.86</td>
<td>29.59</td>
</tr>
<tr>
<td>Moses-chart</td>
<td>32.63</td>
<td>30.05</td>
<td>31.66</td>
</tr>
<tr>
<td>s2t (parse tree)</td>
<td>33.73</td>
<td>30.25</td>
<td>32.75</td>
</tr>
<tr>
<td>s2t (B-parse tree)</td>
<td>34.09</td>
<td>30.99</td>
<td>32.92</td>
</tr>
<tr>
<td>s2t (EM-tree, re-aligned)</td>
<td>34.71</td>
<td>31.55</td>
<td>33.53</td>
</tr>
<tr>
<td>s2t (Bayes-tree)</td>
<td>35.20</td>
<td>31.69</td>
<td>33.89</td>
</tr>
<tr>
<td>Number of sentences</td>
<td>1788</td>
<td>1082</td>
<td>2870</td>
</tr>
</tbody>
</table>
# Experimental Results

<table>
<thead>
<tr>
<th>System</th>
<th>MT04</th>
<th>MT05</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joshua</td>
<td>30.71</td>
<td>27.86</td>
<td>29.59</td>
</tr>
<tr>
<td>Moses-chart</td>
<td>32.63</td>
<td>30.05</td>
<td>31.66</td>
</tr>
<tr>
<td><strong>s2t (parse tree)</strong></td>
<td>33.73</td>
<td>30.25</td>
<td>32.75</td>
</tr>
<tr>
<td>s2t (B-parse tree)</td>
<td>34.09</td>
<td>30.99</td>
<td>32.92</td>
</tr>
<tr>
<td>s2t (EM-tree, re-aligned)</td>
<td>34.71</td>
<td>31.55</td>
<td>33.53</td>
</tr>
<tr>
<td>s2t (Bayes-tree)</td>
<td>35.20</td>
<td>31.69</td>
<td>33.89</td>
</tr>
<tr>
<td>Number of sentences</td>
<td>1788</td>
<td>1082</td>
<td>2870</td>
</tr>
</tbody>
</table>
## Experimental Results

<table>
<thead>
<tr>
<th>System</th>
<th>MT04</th>
<th>MT05</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Joshua</em></td>
<td>30.71</td>
<td>27.86</td>
<td>29.59</td>
</tr>
<tr>
<td><em>Moses-chart</em></td>
<td>32.63</td>
<td>30.05</td>
<td>31.66</td>
</tr>
<tr>
<td><em>s2t (parse trees)</em></td>
<td>33.73</td>
<td>30.25</td>
<td>32.75</td>
</tr>
<tr>
<td><em>s2t (B-parse tree)</em></td>
<td>34.09</td>
<td>30.99</td>
<td>32.92</td>
</tr>
<tr>
<td><em>s2t (EM-tree, re-aligned)</em></td>
<td>34.71</td>
<td>31.55</td>
<td>33.53</td>
</tr>
<tr>
<td><em>s2t (Bayes-tree)</em></td>
<td>35.20</td>
<td>31.69</td>
<td>33.89</td>
</tr>
<tr>
<td>Number of sentences</td>
<td>1788</td>
<td>1082</td>
<td>2870</td>
</tr>
</tbody>
</table>
## Experimental Results

<table>
<thead>
<tr>
<th>System</th>
<th>MT04</th>
<th>MT05</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joshua</td>
<td>30.71</td>
<td>27.86</td>
<td>29.59</td>
</tr>
<tr>
<td>Moses-chart</td>
<td>32.63</td>
<td>30.05</td>
<td>31.66</td>
</tr>
<tr>
<td>s2t (parse tree)</td>
<td>33.73</td>
<td>30.25</td>
<td>32.75</td>
</tr>
<tr>
<td>s2t (B-parse tree)</td>
<td>34.09</td>
<td>30.99</td>
<td>32.92</td>
</tr>
<tr>
<td>s2t (EM-tree, re-aligned)</td>
<td>34.71</td>
<td>31.55</td>
<td>33.53</td>
</tr>
<tr>
<td>s2t (Bayes-tree)</td>
<td>35.20</td>
<td>31.69</td>
<td>33.89</td>
</tr>
<tr>
<td>Number of sentences</td>
<td>1788</td>
<td>1082</td>
<td>2870</td>
</tr>
</tbody>
</table>

+0.61
<table>
<thead>
<tr>
<th>System</th>
<th>MT04</th>
<th>MT05</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Joshua</strong></td>
<td>30.71</td>
<td>27.86</td>
<td>29.59</td>
</tr>
<tr>
<td><strong>Moses-chart</strong></td>
<td>32.63</td>
<td>30.05</td>
<td>31.66</td>
</tr>
<tr>
<td>s2t (parse tree)</td>
<td>33.73</td>
<td>30.25</td>
<td>32.75</td>
</tr>
<tr>
<td>s2t (B-parse tree)</td>
<td>34.09</td>
<td>30.99</td>
<td>32.92</td>
</tr>
<tr>
<td>s2t (EM-tree, re-aligned)</td>
<td><strong>34.71</strong></td>
<td><strong>31.55</strong></td>
<td><strong>33.53</strong></td>
</tr>
<tr>
<td>s2t (Bayes-tree)</td>
<td>35.20</td>
<td>31.69</td>
<td>33.89</td>
</tr>
<tr>
<td>Number of sentences</td>
<td>1788</td>
<td>1082</td>
<td>2870</td>
</tr>
</tbody>
</table>

Experimental Results
Outline

- Introduction
- Tree induction via EM algorithm
- Tree induction via Bayesian inference
- Experiments
- Conclusion
We propose two novel methods to induce effective unsupervised tree structures for tree-based translation.

- Tree Induction via EM algorithm
- Tree Induction via Bayesian inference
Conclusion

The translation results verify that well-designed unsupervised tree structures are actually more appropriate for tree-based translation than parse trees.
Thanks!