Structures in Statistical Machine Translation

Taro Watanabe @ NICT
taro.watanabe@nict.go.jp
Machine Translation

- We learn parameters from data assuming a "model"
- Decode by the learned parameters

The United Inspection Department of Heishantou Port has shortened the procedures for leaving and entering the territory from originally 2 - 3 days to 1 day.
Channel Model

\[ X \xleftarrow{} \text{Process} \xrightarrow{} Y \]
Channel Model
+ noise

\[
\hat{y} = \arg\max_Y Pr(y|x)
\]

\[
= \arg\max_y \frac{Pr(x|y)Pr(y)}{Pr(x)}
\]

\[
= \arg\max_y Pr(x|y)Pr(y)
\]

\[
\hat{e} = \arg\max_e Pr(f|e)Pr(e)
\]

- Employed in: ASR, OCR, MT...
Translation Model

\[ \hat{e} = \underset{e}{\arg \max} \left( \frac{Pr(f|e)}{Pr(e)} \right) \]

Translation Model  Language Model

(Brown et al., 1990)

- Translation Model: adequacy of translation
- Language Model: grammatical correctness, consistent style, fluency
Language Model

\[ Pr(\text{I do not know}) = ? \]
\[ Pr(\text{I not do know}) = ? \]

- Likelihood of a string of English words
- Usually modeled by n-grams

\[ W = w_1, w_2, w_3, \cdots w_N \]
\[ p(W) = p(w_1, w_2, w_3, \cdots, w_N) \]
\[ = p(w_1)p(w_2|w_1)p(w_3|w_1, w_2) \cdots p(w_N|w_1, w_2, w_3, \cdots, w_{N-1}) \]
ngram Language Model

• Markov assumption: only n-word history is memorized

• Bigram:

\[ p(I \text{ do not know}) = p(I)p(\text{do}|I)p(\text{not}|\text{do})p(\text{know}|\text{not}) \]

• Training: Maximum likelihood estimate + smoothing (Good-Turing, Witten-Bell, Kneser-Ney etc.)
Word-based MT

I do not want to work

Je ne veux pas travailler

(Brown et al., 1993)
Phrase-based MT

I do not want to work

Je ne veux pas travailler

(Koehn et al., 2003)
Hierarchical PBMT

I do not want to work

Je ne veux pas travailler

(Chiang, 2007)
Syntax-based MT

I do not want to work

Je ne veux pas travailler

(Galley et al., 2004)
Structures in SMT

- Tutorial
  - Phrase-based MT
  - Tree-based MT
- Syntactic Structures in System Combination
Why Phrases?

• Use phrases as a unit of translations
  • Directly handle many-to-many word correspondence + local reordering
  • Allow local context + non-compositional phrases
• Employed in many systems, including Google, NICT(VoiceTra, TexTra) and open-source, Moses (http://www.statmt.org/moses/)
Phrase-based Model

- Generative story:
  - f is segmented into phrases
  - Each phrase is translated
  - Translated phrases are reordered
Phrase-based Model

\[ \hat{e} = \arg\max_e \frac{\exp (w^\top \cdot h(e, \phi, f))}{\sum_{e', \phi'} \exp (w^\top \cdot h(e', \phi', f))} \]

= \arg\max_e w^\top \cdot h(e, \phi, f)

• Maximization of a log-linear combination of multiple feature functions \( h(e, \Phi, f) \)

• \( \Phi \): phrasal partition of \( f \) and \( e \)

• \( w \): weight of feature functions
Questions

\[ \hat{e} = \arg\max_e w^\top \cdot h(e, \phi, f) \]

- **Training**: How to learn phrases and parameters ($\Phi$ and $h$)?
- **Decoding (or search)**: How to find the best translation (argmax)?
- **Tuning (or optimization)**: How to learn the scaling of features ($w$)?
Training

• Learn phrase pairs from $\mathcal{D} = \langle \mathcal{F}, \mathcal{E} \rangle$

• A standard heuristic approach \cite{Koehn2003} (Koehn et al., 2003)
  • Compute word alignment
  • Extract phrase pairs
  • Score phrases
# Word alignment

<table>
<thead>
<tr>
<th>bushi</th>
<th>yu</th>
<th>shalong</th>
<th>juxing</th>
<th>le</th>
<th>huitan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bush</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>held</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>talk</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sharon</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(Example from Huang and Chiang, 2007)
Extract Phrase Pairs

- From word alignment, extract a phrase pair consistent with word alignment.
**Exhaustive Extraction**

- Exhaustively extract phrases from f, e

<table>
<thead>
<tr>
<th>bushi</th>
<th>yu</th>
<th>shalong</th>
<th>juxing</th>
<th>le</th>
<th>huitan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bush</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>held</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>talk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sharon</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Exhaustively extract phrases from f, e
Features from Phrases

\[
\log p_{\phi}(f|e) = \log \frac{\text{count}(e, f)}{\sum_{f'} \text{count}(e, f')}
\]

\[
\log p_{\phi}(e|f) = \log \frac{\text{count}(e, f)}{\sum_{e'} \text{count}(e', f)}
\]

- Collect all the phrase pairs from the data
- Maximum likelihood estimates by relative frequencies
- Employ scores in two directions
Features from Alignment

\[
\begin{align*}
\log p_{lex}(\bar{f}|\bar{e}, \bar{a}) &= \log \prod_i \frac{1}{|\{j|(i,j) \in \bar{a}\}|} \sum_{\forall (i,j) \in \bar{a}} t(e_i|f_j) \\
\log p_{lex}(\bar{e}|\bar{f}, \bar{a}) &= \log \prod_j \frac{1}{|\{i|(j,i) \in \bar{a}\}|} \sum_{\forall (j,i) \in \bar{a}} t(f_j|e_i)
\end{align*}
\]

- Lexical weighing which scores by word translation probabilities
- Idea: counts for rare phrase pairs are unreliable
- Smoothing effect by decomposing into word pairs
Features for Distortion

- Distance-based distortion modeling

\[ d(f, \phi, e) = | + 2| + |0| + | - 5| = 7 \]
Features for Reordering

- Fine grained reordering features: $\log p_o(o \in \{m, s, d\} | \bar{f}, \bar{e})$
- Either monotone, swap, discontinuous
Other Features

- log of ngram language model(s)
- word count: bias for ngram language model(s)
- phrase count: shorter or longer phrases
Direct Training

• Instead of word alignment + extraction pipeline, directly learn phrase-pairs (Marcu and Wong, 2002)

• Bayesian approach + blocked Gibbs sampling to learn parameters (Blunsom et al., 2009)

• Exhaustively memorize longer phrases (Neubig et al., 2011)
Questions

\[ \hat{e} = \arg\max_{e} w^\top \cdot h(e, \phi, f) \]

- Training: How to learn phrases and parameters (\(\Phi\) and \(h\))?  
- Decoding (or search): How to find the best translation (argmax)?
- Tuning (or optimization): How to learn the scaling of features (\(w\))?
Decoding

\[ \hat{e} = \text{argmax}_{e} \frac{\exp \left( w^\top \cdot h(e, \phi, f) \right)}{\sum_{e', \phi'} \exp \left( w^\top \cdot h(e', \phi', f) \right)} \]

\[ = \text{argmax}_{e} w^\top \cdot h(e, \phi, f) \]

• Given an input sentence f and phrasal model h and w, seek e with the highest score

• Potential errors:
  • Search error: we cannot find the best scored hypothesis
  • Translation error: highest scored hypothesis is bad
Enumerate Phrase Pairs

- Given a input sentence $f$, we can enumerate all possible phrases that match with the source side
- Choose the best phrase pair + ordering
Phrase-based Search Space

- **Node**: bit-vector representing covered source words
- **Edge**: phrasal translations, strictly left-to-right
- **Search space**: $O(2^n)$, Time: $O(2^n n^2)$ (Why?)
Traveling Salesman Problem

- NP-hard problem: visit each city only once
- MT as a Traveling Salesman Problem (Knight, 1999)
  - Each source word corresponds to a city
  - A Dynamic Programming solution:
    - State: visited cities (bit-vector)
    - Search space: $O(2^n)$
- Distortion limit to reduce search space
  i.e. long distortion: \[ \begin{array}{c}
  \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \\
  \end{array} \rightarrow \begin{array}{c}
  \bullet \quad \bullet \quad \bullet \quad \bullet \quad \bullet \\
  \end{array} \]
Non-local features

- Features that requires scoring out of phrases: bigram language model

- Additional state representation required for “future scoring”: I-word for bigram LM

- Space: $O(2^n V^{m-1})$, Time: $O(2^n V^{m-1} n^2)$ for m-gram LM
Phrase-based Decoding

- Re-organize the search space by the cardinality (= # of covered source words)
- Expand hypotheses from the smallest cardinality first
• Prune hypotheses in a bin sharing the same cardinality

• Expand survived hypotheses only
Questions

\[ \hat{e} = \arg\max_e w^\top \cdot h(e, \phi, f) \]

- **Training**: How to learn phrases and parameters (\(\Phi\) and \(h\))?  
- **Decoding (or search)**: How to find the best translation (argmax)?
- **Tuning (or optimization)**: How to learn the scaling of features (\(w\))?
Tuning

\[
\hat{e} = \arg\max_e \frac{\exp (w^\top \cdot h(e, \phi, f))}{\sum_{e',\phi'} \exp (w^\top \cdot h(e', \phi', f))}
= \arg\max_e w^\top \cdot h(e, \phi, f)
\]

- Three popular objectives (in SMT) for tuning \( w \)
  - (Direct) Error Minimization (Och, 2003)
  - Maximum Entropy (Och and Ney, 2002)
  - Large Margin (Watanabe et al., 2007; Chiang et al., 2008; Hopkins and May, 2011)
(Direct) Minimum Error

\[ \hat{w} = \arg\min_w \sum_{s=1}^{S} l(\arg\max_e w^\top \cdot h(e, f_s), e_s) \]

- MERT (Minimum ERror Training)
- Standard in SMT (but not in other NLP areas, such as tagging etc.)
- We can incorporate arbitrary error functions, l
- “Summation” can be replaced by document-wise BLEU specific summation
- 10+ real valued features
n-best Approximation

1: procedure MERT(\{(e_s, f_s)\}_s=1^S)
2: \textbf{for} n = 1...N \textbf{do}
3:   Decode and generate nbest list using w
4:   Merge nbest list
5:  \textbf{for} k = 1...K \textbf{do}
6:    \textbf{for} each parameter m = 1...M \textbf{do}
7:      Solve one dimensional optimization
8:    \textbf{end for}
9:  \textbf{end for}
10: \textbf{end for}
11: \textbf{end procedure}

• N iterations, with each iteration, n-bests are generated and merged

• K iterations, with each iteration, M dimensions are tried (M = # of features), and w is updated
Efficient Line Search

\[ \hat{e} = \arg\max_e w_m^\top \cdot h_m(e, f_s) + w_{m-}^\top \cdot h_{m-}(e, f_s) \]

- If we choose one dimension \( m \), and others fixed, we can treat each hypothesis \( e \) as a “line”

- Compute convex hull of a set of “lines”
In Figure 1, we describe the optimization of error surface for both grid-based and gridless smoothing methods. The unsmoothed error count and smoothed error rate (alpha=3) are computed for a parameter selection. Smoothed error rate is more likely to be better than the unsmoothed error count.

Here, we try to compute the smoothing such that it makes the loss function more probable. This is done by keeping the corpus and its corresponding sequence. Hence, we might find an optimum, which is guaranteed.

In Section 5, we describe the algorithm for smoothed smoothing. The linear error surface (see Figure 2) is used to exhaust the parameter space. Here, we try to score the sentence and find the best candidates.

This section describes how to translate the source language to the target language. However, the results show that our approach to gridless smoothing is more efficient than the grid-based method. Hence, we have implemented an incremental smoothing algorithm (Och, 2003) and evaluated its performance.

(Och, 2003)
MERT in Practice

• Many random starting points (Macherey et al., 2008; Moore and Quirk, 2008)

• Many random directions (Macherey et al., 2008)

• Error count smoothing (Cer et al., 2008)

• Regularization (Hayashi et al., 2009)

• Multi-dimensional search by efficiently computing convex hull (Galley and Quirk, 2011)

• MERT at least 3 times, and report average BLEU (Clark et al., 2011)
Maximum Entropy

\[ \hat{w} = \arg\min_w \frac{\lambda}{2} \|w\|^2 - \sum_{s=1}^{S} \log \frac{\sum_{e^* \in \text{ORACLE}(f_s)} \exp (w^\top \cdot h(e^*, f_s))}{\sum_{e' \in \text{GEN}(f_s)} \exp (w^\top \cdot h(e', f_s))} \]

- Minimize the negative log-likelihood of generating good translations (Och and Ney, 2002)
- ORACLE is a subset of GEN, a set of hypotheses with minimum loss
- Optimized by L-BFGS or SGD
- Potentially large # of features as in NLP tasks
Why Not MaxEnt?

<table>
<thead>
<tr>
<th>error criterion used in training</th>
<th>mWER [%]</th>
<th>mPER [%]</th>
<th>BLEU [%]</th>
<th>NIST</th>
<th># words</th>
</tr>
</thead>
<tbody>
<tr>
<td>confidence intervals</td>
<td>+/- 2.7</td>
<td>+/- 1.9</td>
<td>+/- 0.8</td>
<td>+/- 0.12</td>
<td>-</td>
</tr>
<tr>
<td>MMI</td>
<td>68.0</td>
<td>51.0</td>
<td>11.3</td>
<td>5.76</td>
<td>21933</td>
</tr>
<tr>
<td>mWER</td>
<td>68.3</td>
<td>50.2</td>
<td>13.5</td>
<td>6.28</td>
<td>22914</td>
</tr>
<tr>
<td>smoothed-mWER</td>
<td>68.2</td>
<td>50.2</td>
<td>13.2</td>
<td>6.27</td>
<td>22902</td>
</tr>
<tr>
<td>mPER</td>
<td>70.2</td>
<td>49.8</td>
<td>15.2</td>
<td>6.71</td>
<td>24399</td>
</tr>
<tr>
<td>smoothed-mPER</td>
<td>70.0</td>
<td>49.7</td>
<td>15.2</td>
<td>6.69</td>
<td>24198</td>
</tr>
<tr>
<td>BLEU</td>
<td>76.1</td>
<td>53.2</td>
<td>17.2</td>
<td>6.66</td>
<td>28002</td>
</tr>
<tr>
<td>NIST</td>
<td>73.3</td>
<td>51.5</td>
<td>16.4</td>
<td>6.80</td>
<td>26602</td>
</tr>
</tbody>
</table>

- In Och and Ney (2002), they used
  - WER to select oracle translations
  - n-best merging approach to approximate summation as in MERT
Large Margin

\[ \hat{w} = \arg\min_w \frac{\lambda}{2} \|w\|^2 + \sum_{s=1}^S \sum_{e_s^*} \sum_{e'_s} \xi_{s,e_s^*,e'_s} \]

\[ \begin{align*}
    w^\top \cdot h(e^*_s, f_s) - w^\top \cdot h(e'_s, f_s) & \geq l(e'_s, e^*_s) - \xi_{s,e^*_s,e'_s} \\
    e^*_s & \in \text{ORACLE}(f_s) \\
    e'_s & \in \text{GEN}(f_s)
\end{align*} \]

- Structured output learning approach
- Very hard to enumerate all possible \( e' \) and oracle translations \( e^* \)
- Solution: online learning or n-best approximation
Online Learning

Require: $\{(f_s, e_s)\}_{s=1}^S$

1: $w^1 = \{0\}$
2: $t = 1$
3: for 1...N do
4: $s \sim \text{random}(1, S)$
5: $\hat{e} \in \text{GEN}(f_s, w^{t-1})$
6: if $l(\hat{e}, e_s) \geq 0$ then
7: $w^{t+1} = w^t + h(e_s, f_s) - h(\hat{e}, f_s)$
8: $t = t + 1$
9: end if
10: end for
11: return $w^t$ or $\frac{1}{N} \sum_{i=1}^{N} w^j$

• Averaged perceptron (Liang et al., 2006)

• Scale to large data, but each iteration requires decoding + weight update
Online Large Margin

\[ \hat{w} = \arg\min_{w'} \frac{\lambda}{2} ||w' - w||^2 + \max (l_s - w'^\top \cdot \Delta h_s) \]

\[ \hat{e}_s = \arg\max_e w^\top \cdot h(e, f_s) \]

\[ l_s = l(\hat{e}_s) - l(e^*_s) \]

\[ \Delta h_s = h(\hat{e}_s, f_s) - h(e^*, f_s) \]

- line 7 is replaced by the solution of the above equation
- Still, requires decoding + update in each iteration
- Hard to determine when to stop (watch another dev data)
Ranking Approach

\[ \hat{w} = \arg\min_w \frac{\lambda}{2} \|w\|^2 + \sum_{s=1}^{S} \sum_{e''} \sum_{e'} \xi_{s,e'',e'} \]

\[ - \log \left( 1 + \exp(-w^\top \cdot \Delta h_{e'',e'}) \right) \geq -\xi_{s,e'',e'} \]

\[ e'', e' \in \text{GEN}(f_s) \]

\[ l(e', e'') > 0 \]

\[ \Delta h_{e'',e'} = h(e'', f_s) - h(e', f_s) \]

• An n-best approximation approach (Hopkins and May, 2011)

• Pair-wise comparison of all the hypotheses

• logistic-loss (or 0-1 loss): use an off-the-shelf binary classifier
• Reranking is competitive to MERT and MIRA, and scales to large # of features
Answered?

- Grammar-less model (but very strong)
- Fast decoding
- Why MERT? (Good for non-binary, numerical features)
Structures in SMT

• Tutorial
  • Phrase-based MT
• Tree-based MT
• Syntactic Structures in System Combination
Tree-based MT

• Backgrounds
  • CFG, parsing, hypergraph, deductive system semirings

• Tree-based SMT
  • Synchronous-CFG
  • String-to-Tree, Tree-to-String
Backgrounds: CFG

- Parsing = intersection of CFG with a string (regular grammar)
Parsing: CKY

- $O(n^3)$: For each length $n$, for each position $i$, for each rule $X \rightarrow YZ$, for each split point $k$

- (Bottom-up) topological order
Hypergraph

- Generalization of graphs:
  - $h(e)$: head node of hyperedge $e$
  - $T(e)$: tail node(s) of hyperedge $e$, arity = $|T(e)|$
  - hyperedge = instantiated rule
- Represented as and-or graphs

(Klein and Manning, 2001)
Deductive System

- Parsing algorithm as a deductive system
- We start from initial items (axioms) until we reach a goal item
- If antecedents are proved, its consequent is proved
- deduction = hyperedge

\[ VP_{1,6} \rightarrow VBD_{1,2} \rightarrow NP_{2,6} \rightarrow VP_{i,j} \rightarrow VBZ_{j,k} \rightarrow NP_{i,k} \]

(Shieber et al., 1995)
A polynomial space encoding of exponentially many parses by sharing common sub-derivations

Single derivation = tree

(Klein and Manning, 2001; Huang and Chiang, 2005)
## Summary of Formalisms

<table>
<thead>
<tr>
<th>hypergraph</th>
<th>AND/OR graph</th>
<th>CFG</th>
<th>deductive system</th>
</tr>
</thead>
<tbody>
<tr>
<td>vertex</td>
<td>OR-node</td>
<td>symbol</td>
<td>item</td>
</tr>
<tr>
<td>source-vertex</td>
<td>leaf OR-node</td>
<td>terminal</td>
<td>axiom</td>
</tr>
<tr>
<td>target-vertex</td>
<td>root OR-node</td>
<td>start symbol</td>
<td>goal item</td>
</tr>
<tr>
<td>hyperedge</td>
<td>AND-node</td>
<td>production</td>
<td>instantiated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>deduction</td>
</tr>
</tbody>
</table>

$$\langle v, \{u_1, u_2\} \rangle$$

$$v \rightarrow u_1 u_2$$

$$\frac{u_1 u_2}{v}$$
Weights and Semirings

\[ \begin{align*}
\text{VP} & \xrightarrow{w_1} \text{VBD NP} \\
\text{NP} & \xrightarrow{w_2} \text{NP PP}
\end{align*} \]

\[ \begin{align*}
\text{VP}_{1,6} : w_1 \otimes c \otimes d \\
\text{VBD}_{1,2} : c & \quad \text{NP}_{2,6} : d \\
\text{NP}_{2,6} : w_2 \otimes a \otimes b \\
\text{NP}_{2,4} : a & \quad \text{PP}_{4,6} : b \\
\text{NP}_{2,6} : w_2 \otimes a \otimes b
\end{align*} \]

• Associate weights as in WFST

• \( \otimes \) : extension (multiplicative), \( \oplus \) : summary (additive)
Weights and Semirings

- The weight of a hyperedge is dependent on antecedents (non-monotonic)
- The weight of a derivation is the product of hyperedge weights
- The weight of a vertex is the summary of (sub-)derivation weights

\[
d(v) = (w(e_1, u_1, u_2) \otimes d(u_1) \otimes d(u_2)) \oplus (w(e_2, u_3, u_4) \otimes d(u_3) \otimes d(u_4))
\]
Semirings

\[ K = \langle K, \oplus, \otimes, 0, 1 \rangle \]

<table>
<thead>
<tr>
<th>semiring</th>
<th>(K)</th>
<th>(\oplus)</th>
<th>(\otimes)</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viterbi</td>
<td>[0, 1]</td>
<td>max</td>
<td>\times</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Real</td>
<td>(\mathbb{R})</td>
<td>+</td>
<td>\times</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Log</td>
<td>(\mathbb{R})</td>
<td>logsumexp</td>
<td>+</td>
<td>+\infty</td>
<td>0</td>
</tr>
<tr>
<td>Tropical</td>
<td>(\mathbb{R})</td>
<td>min</td>
<td>+</td>
<td>+\infty</td>
<td>0</td>
</tr>
<tr>
<td>Expectation</td>
<td>(\langle P, R \rangle)</td>
<td>(\langle p_1 \oplus p_2, r_1 \oplus r_2 \rangle)</td>
<td>(\langle p_1 \otimes p_2, p_1 \otimes r_2 \oplus p_2 \otimes r_1 \rangle)</td>
<td>\langle 0, 0 \rangle</td>
<td>\langle 1, 0 \rangle</td>
</tr>
</tbody>
</table>
Conclusion

• Review important concepts from “parsing”
  • CFG, parsing, hypergraph, deductive system, weights, semirings
Tree-based MT

- Backgrounds
  - CFG, parsing, hypergraph, deductive system semirings

- Tree-based SMT
  - Synchronous-CFG
  - String-to-Tree, Tree-to-String
Synchronous-CFG

\[ \hat{e} = \arg\max_e \frac{\exp \left( \mathbf{w}^\top \cdot \mathbf{h}(e, D, f) \right)}{\sum_{e', D'} \exp \left( \mathbf{w}^\top \cdot \mathbf{h}(e', D', f) \right)} \]

\[ = \arg\max_e \mathbf{w}^\top \cdot \mathbf{h}(e, D, f) \]

- D: a single derivation constructed by intersecting SCFG with input string
Synchronous-CFG: Model

$$S \rightarrow \langle S_1 \ X_2, S_1 \ X_2 \rangle$$

$$S \rightarrow \langle X_1, X_1 \rangle$$

$$X \rightarrow \langle X_1 \text{举行} \ X_2, \text{hold} \ X_2 \ X_1 \rangle$$

$$X \rightarrow \langle \text{与 沙龙}, \text{with Sharon} \rangle$$

$$VP \rightarrow \langle \text{VBD}_1 \ \text{NP}_2, \ NP_2 \ \text{VBD}_1 \rangle$$

$$NP \rightarrow \langle \text{NP}_1 \ \text{PP}_2, \ NP_1 \ \text{PP}_2 \rangle$$

$$VP \rightarrow \langle \text{VBD}_1 \ \text{NP}_2 \ \text{PP}_3, \ NP_2 \ \text{PP}_3 \ \text{VBD}_1 \rangle$$

- We use two categories, S and X (Chiang, 2007)

- Or, borrow linguistic categories from syntactic parse (Zollman and Venugopal, 2006)
Rule Extraction

布什 与 沙龙举行了会谈

- As in phrase-based models, extract phrases then, use sub-phrases as non-terminals, aka Hiero (Chiang, 2007)

(Example from Huang and Chiang, 2007)
• Borrow syntactic categories either from source/target side, aka SAMT (Zollman and Venugopal, 2006)
Exhaustive Extraction

布什 与 沙龙举行 了 会谈

X₁ X₂ 了 会谈  X₂ a talk X₁
X₁ X₂ 会谈  X₂ a talk X₁
X₁ X₂ 举行 X₂  X₂ held X₁ X₁
X₁ X₂ 举行 了 X₂  X₂ held a X₁ X₁
X₁ X₂ 与 与 Xₙ X₂ X₁ X₁ with Sharon
X₁ X₂ 与 X₁ X₂ X₁ with X₁

S → ⟨S₁ X₂, S₁ X₂⟩
S → ⟨X₁, X₁⟩

- Exhaustively extract rules as in phrase-based MT
- + glue rules
Features from Rules

\[
\log p_r(\bar{\alpha}|\bar{\beta}) = \log \frac{\text{count}(\bar{\beta}, \bar{\alpha})}{\sum_{\bar{\alpha}'} \text{count}(\bar{\beta}, \bar{\alpha}')} \\
\log p_r(\bar{\beta}|\bar{\alpha}) = \log \frac{\text{count}(\bar{\beta}, \bar{\alpha})}{\sum_{\bar{\beta}'} \text{count}(\bar{\beta}', \bar{\alpha})}
\]

- Collect all the rules \((\alpha, \beta)\) from the data:
  - \(\alpha = \) source side string, \(\beta = \) target side string
- Maximum likelihood estimates by relative frequencies
- Employ scores in two directions
Remarks on Rules

- Too many rules extracted (Chiang, 2007):
  - at most two non-terminal symbols
  - at least one terminal between non-terminals in the source side
  - Span at most 15 words for “holes”

- Fractional counts (Chiang, 2007):
  - Each phrases counted in phrase-based MT
  - Fractional counts for rules sharing the same source/target span
Other Features

- Lexical weights as used in phrase-based MT
- ngram language model(s)
- word count: bias for ngram language model(s)
- rule count: shorter or longer phrases
- glue-rule counts: bias for monotonic glue rules
Parse input sentence using the source side, and construct a translation forest by target side.
Synchronous-CFG: Parsing

- Translation by SCFG = monolingual parsing using the source side grammar
- Construct forest by the projected target side
- From forests, compute the best derivation (Huang and Chiang, 2005)
- Complexity: $O(n^3)$ as in monolingual CKY
Non-Local Features

\[ X \rightarrow \langle X_{1} \text{juxing} X_{2}, \text{held} X_{2} X_{1} \rangle \]

\[ X_{1,6} \]

- \[ p(\text{talk} \mid a) \]
  - a talk
  - talks
  - meeting
  - meetings

- \[ p(\text{Sharon} \mid \text{with}) \]
  - held a talk with Sharon
  - held talks with Sharon
  - held a talk and Sharon
  - held meeting Sharon with

Update boundary words only

- \[ p(\text{Sharon} \mid \text{and}) \]
  - held Sharon with
  - Sharon and

- \[ p(\text{with} \mid \text{Sharon}) \]
  - p(with | Sharon)

- \[ p(\text{and} \mid \text{Sharon}) \]
  - p(and | Sharon)

- non-local features which requires out-of-span context, i.e. bigram LM

\[ \text{DmtBM; s} \]
\[ \text{s} \]
\[ \text{s} \]
\[ \text{1} \]
\[ \text{2} \]
\[ \text{s} \]
\[ \text{2} \]
Bigram Features

\[ X \rightarrow \langle X_1 \text{juxing} X_2, \text{held} X_2 X_1 \rangle \]

- \( X_{1,6} \)
- \( X_{1,3} \)
- \( X_{3,4} \)

- \( \text{a * talk} \)
- \( \text{talks} \)
- \( \text{meeting} \)
- \( \text{meetings} \)
- \( \text{held * Sharon} \)
- \( \text{held * Sharon} \)
- \( \text{held * with} \)
- \( \text{held * Sharon} \)
- \( \text{held * Sharon} \)
- \( \text{held * with} \)
- \( \text{with * Sharon} \)
- \( \text{and * Sharon} \)
- \( \text{Sharon * with} \)
- \( \text{Sharon * and} \)

• We keep only bigram states: (Why 2 words?)
Language Model Updates

• Each hypothesis keeps two contexts:
  • Prefix: ngrams to be scored with antecedents
  • Suffix: contexts for future ngrams (i.e. Phrase-based MT)
• Complexity: $O(n^3V^{2(m-1)})$
• Very inefficient: we need to explicitly enumerate all the hypotheses in antecedents
Forest Rescoring

• Translation by SCFG = monolingual parsing using the source side grammar

• Construct forest by the projected target side + Rescore with non-local features

• From forests, compute the best derivation (Huang and Chiang, 2005)

• Complexity: $O(n^3)$ as in monolingual CKY
Cube Pruning

\[ X \rightarrow \langle X_{\downarrow 1} \text{juxing } X_{\downarrow 2}, \text{ held } X_{\downarrow 2} X_{\downarrow 1} \rangle \]

<table>
<thead>
<tr>
<th></th>
<th>with * Sharon</th>
<th>and * Sharon</th>
<th>Sharon * with</th>
<th>Sharon * and</th>
</tr>
</thead>
<tbody>
<tr>
<td>a * talk</td>
<td>1.0</td>
<td>2.5</td>
<td>2.7</td>
<td>3.6</td>
</tr>
<tr>
<td>talks</td>
<td>1.3</td>
<td>2.8</td>
<td>3.0</td>
<td>3.9</td>
</tr>
<tr>
<td>meeting</td>
<td>2.2</td>
<td>3.7</td>
<td>3.9</td>
<td>4.8</td>
</tr>
<tr>
<td>meetings</td>
<td>2.6</td>
<td>4.1</td>
<td>4.3</td>
<td>5.2</td>
</tr>
</tbody>
</table>

- For each hyperedgedge, create a “cube” representing combinations of antecedents (Huang and Chiang, 2007)
Cube Pruning

X → ⟨X₁ juxing X₂, held X₂ X₁⟩

<table>
<thead>
<tr>
<th>Item</th>
<th>Sharon with *</th>
<th>Sharon and *</th>
<th>Sharon * with</th>
<th>Sharon * and</th>
</tr>
</thead>
<tbody>
<tr>
<td>a * talk</td>
<td>1.0</td>
<td>2.5 (+0.5)</td>
<td>2.7 (+1.0)</td>
<td>3.6 (+1.5)</td>
</tr>
<tr>
<td>talks</td>
<td>1.3</td>
<td>2.8 (+0.3)</td>
<td>3.0 (+1.5)</td>
<td>3.9 (+2.0)</td>
</tr>
<tr>
<td>meeting</td>
<td>2.2</td>
<td>3.7 (+0.5)</td>
<td>3.9 (+1.0)</td>
<td>4.8 (+1.5)</td>
</tr>
<tr>
<td>meetings</td>
<td>2.6</td>
<td>4.1 (+0.3)</td>
<td>4.3 (+1.5)</td>
<td>5.2 (+2.0)</td>
</tr>
</tbody>
</table>

- Bigrams require contexts from antecedents: non-monotonic scoring
Cube Pruning

queue: (0,0)
k-best:

<table>
<thead>
<tr>
<th></th>
<th>with * Sharon</th>
<th>and * Sharon</th>
<th>Sharon * with</th>
<th>Sharon * and</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| a * talk | 1.0 | 3.0 |
| talks    | 1.3  |     |
| meeting  | 2.2  |     |
| meetings | 2.6  |     |

- Starting from the upper-left corner, enumerate antecedent combinations
Cube Pruning

queue:
k-best: (0,0)

<table>
<thead>
<tr>
<th></th>
<th>with * Sharon</th>
<th>and * Sharon</th>
<th>Sharon * with</th>
<th>Sharon * and</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>1.7</td>
<td>2.6</td>
<td>3.2</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td></td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Starting from the upper-left corner, enumerate antecedent combinations
Cube Pruning

- Starting from the upper-left corner, enumerate antecedent combinations

<table>
<thead>
<tr>
<th></th>
<th>with * Sharon</th>
<th>and * Sharon</th>
<th>Sharon * with</th>
<th>Sharon * and</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td></td>
<td></td>
<td>2.6</td>
<td>3.2</td>
</tr>
<tr>
<td>1.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.0</td>
<td>3.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a * talk</td>
<td>1.0</td>
<td>3.0</td>
<td>3.7</td>
<td></td>
</tr>
<tr>
<td>talks</td>
<td>1.3</td>
<td>3.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>meeting</td>
<td>2.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>meetings</td>
<td>2.6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cube Pruning

queue: (1,0)
k-best: (0,0)(0,1)

<table>
<thead>
<tr>
<th></th>
<th>with * Sharon</th>
<th>and * Sharon</th>
<th>Sharon * with</th>
<th>Sharon * and</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a * talk</td>
<td>1.0</td>
<td>3.0</td>
<td>3.7</td>
<td></td>
</tr>
<tr>
<td>talks</td>
<td>1.3</td>
<td>3.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>meeting</td>
<td>2.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>meetings</td>
<td>2.6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Starting from the upper-left corner, enumerate antecedent combinations
Cube Pruning

queue: (1,0)(0,2)(1,1)
k-best: (0,0)(0,1)

<table>
<thead>
<tr>
<th></th>
<th>with * Sharon</th>
<th>and * Sharon</th>
<th>Sharon * with</th>
<th>Sharon * and</th>
</tr>
</thead>
<tbody>
<tr>
<td>a * talk</td>
<td>1.0</td>
<td>3.0</td>
<td>3.7</td>
<td></td>
</tr>
<tr>
<td>talks</td>
<td>1.3</td>
<td>3.1</td>
<td>4.5</td>
<td></td>
</tr>
<tr>
<td>meeting</td>
<td>2.2</td>
<td>4.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>meetings</td>
<td>2.6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Starting from the upper-left corner, enumerate antecedent combinations
Cube Pruning

queue: \((0,2) (1,1)\)

k-best: \((0,0) (0,1) (1,0)\)

<table>
<thead>
<tr>
<th></th>
<th>with * Sharon</th>
<th>and * Sharon</th>
<th>Sharon * with</th>
<th>Sharon * and</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>1.7</td>
<td>2.6</td>
<td>3.2</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>3.0</td>
<td>3.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td>3.1</td>
<td>4.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td></td>
<td>4.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Starting from the upper-left corner, enumerate antecedent combinations
## Cube Pruning

**queue:** (0,2) (1,1) (3,0)  
**k-best:** (0,0) (0,1) (1,0)

Starting from the upper-left corner, enumerate antecedent combinations.

<table>
<thead>
<tr>
<th></th>
<th>1.5</th>
<th>1.7</th>
<th>2.6</th>
<th>3.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a * talk</td>
<td>1.0</td>
<td>3.0</td>
<td>3.7</td>
<td>5.1</td>
</tr>
<tr>
<td>talks</td>
<td>1.3</td>
<td>3.1</td>
<td>4.5</td>
<td></td>
</tr>
<tr>
<td>meeting</td>
<td>2.2</td>
<td>4.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>meetings</td>
<td>2.6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Starting from the upper-left corner, enumerate antecedent combinations.
Cube Pruning

queue: \((1,1)(3,0)\)
k-best: \((0,0)(0,1)(1,0)(0,2)\)

<table>
<thead>
<tr>
<th></th>
<th>with Sharon</th>
<th>and * Sharon</th>
<th>Sharon * with</th>
<th>Sharon * and</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| a * talk | 1.0  | 3.0  | 3.7  | 5.1  |
| talks    | 1.3  | 3.1  | 4.5  |      |
| meeting  | 2.2  | 4.2  |      |      |
| meetings | 2.6  |      |      |      |

- Starting from the upper-left corner, enumerate antecedent combinations
Cube Pruning

queue: (0,4) (1,1) (1,2) (3,0)
k-best: (0,0) (0,1) (1,0) (0,2)

<table>
<thead>
<tr>
<th></th>
<th>1.5</th>
<th>1.7</th>
<th>2.6</th>
<th>3.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a * talk</td>
<td>1.0</td>
<td>3.0</td>
<td>3.7</td>
<td>5.1</td>
</tr>
<tr>
<td>talks</td>
<td>1.3</td>
<td>3.1</td>
<td>4.5</td>
<td></td>
</tr>
<tr>
<td>meeting</td>
<td>2.2</td>
<td>4.2</td>
<td>4.9</td>
<td></td>
</tr>
<tr>
<td>meetings</td>
<td>2.6</td>
<td>4.4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Starting from the upper-left corner, enumerate antecedent combinations
Multiple Rules

- Multiple rules sharing the same span are queued
- Each rule is associated with a cube
- hypothesis = hyperedge + cube-position
Further Faster Pruning

- Cube Growing (Huang and Chiang, 2007)
  - Top-down pruning combined with heuristic estimates
- Faster Cube Pruning (Gesmundo and Henderson, 2010)
  - Eliminate bookkeeping for inserted hypotheses by determining the ordering of cube enumerations
  - Push minimum hypotheses by looking up ancestors
- Incremental (Huang and Mi, 2010)
  - Top-down decoding as in (Watanabe et al., 2006)
Conclusion

• Synchronous-CFG
  • paired CFG + shared non-terminal symbols
• Training is based on phrase-based MT by treating sub-phrase as a non-terminal
• Decoding: monolingual parsing
  • An efficient antecedent combination via cube-pruning
Tree-based MT

- Backgrounds
  - CFG, parsing, hypergraph, deductive system semirings
- Tree-based SMT
  - Synchronous-CFG
- String-to-Tree, Tree-to-String
Each synchronous rule has a subtree structure

Flat structure + sharing the same non-terminal symbols = synchronous-CFG
Tree-to-String Rules

PP
\[\text{P} \rightarrow \text{NPB} \]
\[\text{yu shalong} \rightarrow \text{with Sharon} \]

VPB
\[\text{VS} \rightarrow \text{AS} \rightarrow x_1:NPB \]
\[\text{juxing le} \rightarrow \text{held a } x_1 \]

QP
\[x_1:CD \rightarrow \text{CLP} \rightarrow \text{ben} \rightarrow x_1 \]

IP
\[x_1:NP \rightarrow \text{VP} \rightarrow x_2:IP \rightarrow x_3:VPB \rightarrow x_1:NP \rightarrow \text{DEG} \rightarrow \text{de} \rightarrow x_2 \text{ of } x_1 \]
ルールの抽出

• Compute “minimum rules” as in phrase-based MT

(Galley et al., 2004)
Rule Extraction

IP(0,1,3,4,5)

NPB(0)  VP (1,3,4,5)

bushi  PP(4,5)  VPB(1,3)

P(4) NPB(5)  VS(1) AS(1) NPB(3)

yu shalong juxing le huitan

Bush held a talk with Sharon

* Compute “spans” by propagating alignment in bottom-up

(Galley et al., 2004)
Rule Extraction

- Compute “complements” in top-down
• Compute “frontiers”: The nodes in which the intersection of “spans” and “complements” is empty.
• Extract minimum rules using frontiers
Rule Extraction

- Extract minimum rules using frontiers

Example:
NPB
| bushi
→ Bush

NPB
| bushi
→ Bush

yu shalong juxing le huitan
Bush held a talk with Sharon

• Extract minimum rules using frontiers
• Extract minimum rules using frontiers
• Extract “compound rules” by combining minimum rules (i.e. longer phrases)
Decoding: String-\{String, Tree\}

- Similar to SCFG decoding: Use the “collapsed” source side rule to perform CKY parsing

- Construct a translation forest using the target side

\[
\begin{align*}
\langle \text{VPB} \rangle & \rightarrow \text{juxing le NPB}_1, \\
\langle \text{NP} \rangle & \rightarrow \text{NP}_1 \text{ de NP}_2, \\
x & \rightarrow \text{hold a } x_1, \\
x & \rightarrow x_2 \text{ of } x_1
\end{align*}
\]

(Galley et al., 2004)
Decoding: Tree-\{String, Tree\}

- First, an input sentence is parsed
- Input tree is transformed into a translation forest by tree rewriting

(Huang et al., 2006)
Forest Rescoring

• Translation by \{tree,string\}-to-\{tree,string\}
  • string-to-\{tree,string\}: parsing using the source-side grammar
  • tree-to-\{tree,string\}: parse input sentences + tree-match-rewrite

• Construct forest by the projected target side

• From forests, compute the best derivation (Huang and Chiang, 2005)
Conclusion

- {String, Tree}-to-{String, Tree} translation models
- Rules extraction by GHKM (Galley et al., 2004)
  - Galley M, Hopkins M, Knight K, Marcu D, 2004
- Decoding:
  - String-to-{String, Tree} by CKY
  - Tree-to-{String, Tree} by tree-rewrite
More on Tree-based Models

- Forest-based approach: instead of 1-best parse, use forest encoding k-bests (Mi and Huang, 2008; Mi et al., 2008)
- “Binarized forest” as an alternative to represent multiple parses (Zhang et al., 2011)
- Fuzzy tree-to-tree as a way to overcome “strictness” of tree-based models (Chiang, 2010)
- Use of dependency (Mi and Liu, 2010; Xie et al., 2011)
Tree-based MT

- Backgrounds
  - CFG, parsing, hypergraph, deductive system semirings
- Tree-based SMT
  - Synchronous-CFG
  - String-to-Tree, Tree-to-String
Structures in SMT

- Tutorial
  - Phrase-based MT
  - Tree-based MT

- Syntactic Structures in System Combination
MT System Combination by Confusion Forest

Taro Watanabe and Eiichiro Sumita
@ NICT
MT System Combination

- Better translation by combining multiple system outputs:
  - Sentence selection (Nomoto, 2004; etc.)
  - Phrasal combination (Frederking and Nirenburg, 1994; etc.)
  - Word level combination (Bangalore et al., 2001; Matusov et al., 2006; etc.)
- This Work: Syntactic combination, not word-wise combination
Confusion Network

★ I saw the forest
I walked the blue forest
I saw the green trees
the forest was found

- State-of-the-art: Confusion Network
- Choose a skeleton, compute word alignment against the skeleton
  - Edit-distance-based alignment (TER etc.) (Sim et al., 2007)
  - Model-based alignment (GIZA++ etc.) (Matsoy et al., 2006)
• Construct a network with each arc representing alternative translation

• Best path = Best translation

• Syntactically different language pairs: i.e. active/passive voices

• Spurious insertion/repetition due to alignment error

• Incremental alignment/construction + merge multiple networks into one (Rosti et al., 2008)
Confusion Forest

- Compactly represent multiple parses by sharing nodes
- Represented by “hypergraph”
• Parse each system output by a parser

• Extract rules from parsed trees: local grammar
Generation by Earley

Scan:

$\frac{[X \to \alpha \cdot x\beta, h] : u}{[X \to \alpha x \cdot \beta, h] : u}$

Predict:

$\frac{[X \to \alpha \cdot Y\beta, h]}{[Y \to \bullet \gamma, h + 1] : u}$

$Y \xrightarrow{u} \gamma \in \mathcal{G}, h < H$

Complete:

$\frac{[X \to \alpha \cdot Y\beta, h] : u}{[X \to \alpha Y \cdot \beta, h] : u \otimes v}$

$\frac{[Y \to \gamma\bullet, h + 1] : v}{[X \to \alpha Y \cdot \beta, h] : u \otimes v}$

- Generation from the extracted grammar
- Scanning always succeed: constraint by height
Generation by Earley
Spurious Ambiguity

- Memorize the (partial) tree structures in each node
- Employ the sequence of Earlye state as a node
- Horizontal/Vertical Markovization (Klein and Manning, 2003)
Forest Reranking

\[ \hat{d} = \arg \max_{d \in D} w^\top \cdot h(d, F) \]

- Choose the best derivation \( d \) among all possible derivations \( D \) in a forest \( F \)
- Terminal yield of the best derivation = the best translation
- Approximately apply non-local features (ngram language models) by Cube Pruning (Huang and Chiang, 2007)
- Efficient \( k \)-best by Algorithm 3 (Huang and Chiang, 2005)
Experiments

- WMT10 System Combination Task
- Czech, German, Spanish, French → English
- tune/test: 455/2,034 sentences

<table>
<thead>
<tr>
<th></th>
<th>cz-en</th>
<th>de-en</th>
<th>es-en</th>
<th>fr-en</th>
</tr>
</thead>
<tbody>
<tr>
<td>systems</td>
<td>6</td>
<td>16</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>tune</td>
<td>10.6K</td>
<td>10.9K</td>
<td>10.9K</td>
<td>11.0K</td>
</tr>
<tr>
<td>test</td>
<td>50.5K</td>
<td>52.1K</td>
<td>52.1K</td>
<td>52.4K</td>
</tr>
</tbody>
</table>
Systems

• CF: Stanford parser + “cicada” (a hypergraph-based toolkit based on SEMIring parsing framework)

• CN: Single network by merging multiple networks + conversion into hypergraph by lattice parsing

• features: tuned by hypergraph-MERT(Kumar et al. 2009)
  • Language Models, # of terminals, # of hyperedges
  • # of rules in a derivation originally in n_th system output
  • BLEUs by treating each system output as a reference translation
  • Network distance (only used for CN)
## BLEU

<table>
<thead>
<tr>
<th></th>
<th>cz-en</th>
<th>de-en</th>
<th>es-en</th>
<th>fr-en</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>system min</strong></td>
<td>14.09</td>
<td>15.62</td>
<td>21.79</td>
<td>16.79</td>
</tr>
<tr>
<td><strong>max</strong></td>
<td>23.44</td>
<td>24.10</td>
<td>29.97</td>
<td>29.17</td>
</tr>
<tr>
<td><strong>CN</strong></td>
<td>23.70</td>
<td>24.09</td>
<td>30.45</td>
<td>29.15</td>
</tr>
<tr>
<td>CF,v=\infty,h=\infty</td>
<td>24.13</td>
<td>24.18</td>
<td>30.41</td>
<td>29.57</td>
</tr>
<tr>
<td>CF,v=\infty,h=2</td>
<td>24.14</td>
<td>24.58</td>
<td>30.52</td>
<td>28.84</td>
</tr>
<tr>
<td>CF,v=\infty,h=1</td>
<td>24.01</td>
<td>23.91</td>
<td>30.46</td>
<td>29.32</td>
</tr>
</tbody>
</table>
## Oracle BLEU

<table>
<thead>
<tr>
<th></th>
<th>cz-en</th>
<th>de-en</th>
<th>es-en</th>
<th>fr-en</th>
</tr>
</thead>
<tbody>
<tr>
<td>rerank</td>
<td>29.40</td>
<td>32.32</td>
<td>36.83</td>
<td>36.59</td>
</tr>
<tr>
<td>CN</td>
<td>38.52</td>
<td>34.97</td>
<td>47.65</td>
<td>46.37</td>
</tr>
<tr>
<td>CF,v=∞,h=∞</td>
<td>30.51</td>
<td>34.07</td>
<td>38.69</td>
<td>38.94</td>
</tr>
<tr>
<td>CF,v=∞,h=2</td>
<td>30.61</td>
<td>34.25</td>
<td>38.87</td>
<td>39.10</td>
</tr>
<tr>
<td>CF,v=∞,h=1</td>
<td>31.09</td>
<td>34.65</td>
<td>39.27</td>
<td>39.51</td>
</tr>
</tbody>
</table>
Hypergraph size

<table>
<thead>
<tr>
<th></th>
<th>cz-en</th>
<th>de-en</th>
<th>es-en</th>
<th>fr-en</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN</td>
<td>2,222.68</td>
<td>47,231.20</td>
<td>2,932.24</td>
<td>11,969.40</td>
</tr>
<tr>
<td>CF,v=∞,h=1</td>
<td>230.08</td>
<td>540.03</td>
<td>262.30</td>
<td>386.79</td>
</tr>
<tr>
<td>CF,v=5,h=1</td>
<td>254.45</td>
<td>651.10</td>
<td>302.01</td>
<td>477.51</td>
</tr>
<tr>
<td>CF,v=4,h=1</td>
<td>286.01</td>
<td>802.79</td>
<td>349.21</td>
<td>575.17</td>
</tr>
</tbody>
</table>

- Average # of hyperedges
- (rough) estimates for speed
Conclusion

• System combination by Confusion Forest which employs syntactic distance, not word-level distance

• Forest construction by the grammar extracted from system outputs

• Parser: assign tree structure to the similar expressions

• Compact date structure + comparable performance against Confusion Network
Structures in SMT

• Tutorial
  • Phrase-based MT
  • Tree-based MT
• Syntactic Structures in System Combination
Research on MT

• Reading: at least 50 papers are related to MT “every year”

• Specialist: solve a sub-problem

• Language Neutral: a solution which works only for a particular language pair is boring