Foundations of Statistical Machine Translation: Past, Present and Future

Taro Watanabe
taro.watanabe @ nict.go.jp
http://mastarpj.nict.go.jp/~t_watana/
• Statistical Machine Translation (SMT) started from Brown et al. (1990)

• Is SMT matured?
  • Real service: Web-based (Google, Microsoft), mobile phone (NICT)

• Promising gains from Tree-based approaches
  • Syntax-based SMT in \{tree, string\}-to-\{tree, string\}
  • Decoding = Parsing

• Better model, better search and better training
Statistical Machine Translation?

• MT as a decision making process:
  • Given a source text, search for the best translation

• Difference from Rule-based (Knowledge-based) MT:
  • Learn model/parameters from data

• Difference from Example-based MT:
  • Both are empirical, but more emphasis on examples + (usually) greedy search + heuristics
Overview of overview

• Foundation
  • Model, Training, Decoding
  • Phrase-based SMT
• Tree-based SMT
• Advanced Topics
Foundation
Overview

• Model, Training, Decoding
• Word Alignment
• Phrase-based SMT
• Evaluation
• Optimization
Translation as a decision problem

• Modeling:
  • Good $p(e|f)$ approximating $Pr(e|f)$
  • Linguistic clues will be helpful

• Training:
  • Assign parameters given data
  • Maximum-likelihood, EM-algorithms, Bayesian

• Search:
  • Find the best translation
  • DP-based search with heuristic pruning
Source Channel Model

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

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

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

\[ = \arg\max_e p(f|e)p(e) \]

• Early statistical machine translation (Brown et al., 1990)

• Since we do not know true distribution, we will approximate Pr(f|e) by p(f|e)
Source Channel Model

- Translation Model: $p(f|e)$
  - Bilingual correspondence between two sentences, $f$ and $e$
  - Usually encode linguistic clues, such as dictionary
- Language Model: $p(e)$
  - “fluency” for the generated sentence
Log-linear Model

\[ p(e|f) = \frac{\exp (w \cdot h(e, f))}{\sum_{e'} \exp (w \cdot h(e', f))} \]

- Generalization of Source Channel model
- Each feature function captures different aspect of translations
- Each feature function is weighted
- Easy to incorporate new features
Overview

- Model, Training, Decoding
- Word Alignment
- Phrase-based SMT
- Evaluation
- Optimization
Word alignment

- One of the fundamental unit of translation
- one-to-one correspondence
- or, many-to-many alignment
Word alignment models

- **IBM Model 4**
- Decompose into several models: fertility, lexicon, distortion
Word alignment models

\[ p(a, f | e) = \sum_{j=1}^{J} p_d(a_j | a_{j-}, j) p_t(f_j | e_{a_j}) \]

\[ p_d(a_j = 0 | a_{j-} = i) = p_0 \]

\[ p_d(a_j = i' \neq 0 | a_{j-} = i) \propto (1 - p_0) \begin{cases} 1 & \text{(IBM 1)} \\ c(i' - \lfloor \frac{j}{J} \rfloor) & \text{(IBM 2)} \\ c(i' - i) & \text{(HMM)} \end{cases} \]

- IBM 1, IBM 2 and HMM
- More models, such as IBM \{3,4,5\}
Word alignment training

... その 箱 ... 箱 を 開ける ... 扉 を 開ける ...

... the box ... open the box ... open the door ...

• EM algorithm:
  • E-step to compute expected counts
  • M-step to perform maximization
Word alignment training

• Starting from uniform parameter, try compute expectation of aligning words

• Based on the expectation, estimate parameters

• Iterate....until convergence
Word alignment model training

\[ \hat{\theta} = \arg\max_{\theta} \prod_{e,f} p(f, a|e; \theta) \]

\[ = \arg\max_{\theta} \sum_{e,f} \log p(f, a|e; \theta) \]

E-step: \[ q(a; f, e) = p(a|e, f; \theta) \]

M-step: \[ \theta' = \arg\max_{\theta} \sum_{f,e,a} q(a; f, e) \log p(f, e, a; \theta) \]

- Inside EM-training

- Maximizing log-likelihood over the training data
Alignment combination

- IBM Models are limited to one-to-many
- Prone to errors, especially for rare words
- Training in both directions, “heuristically” combine
Alignment heuristics

- Starts from intersected alignment, greedily add union alignments
Symmetric training

E-step: \( q(a; f, e) = \frac{1}{Z_{f,e}} p_1(a|f, e; \theta_1) \cdot p_2(a|e, f; \theta_2) \)

M-step: \( \theta' = \arg\max_{\theta} \sum_{f,e,a} q(a; f, e) \log p_1(f, e, a; \theta_1) + \sum_{f,e,a} q(a; f, e) \log p_2(f, e, a; \theta_2) \)

(Liang et al., 2006)

• Alternatives to heuristic approaches, it is possible to approximate symmetrization during EM-algorithm

• Jointly maximize both directions by approximating summation (Liang et al., 2006)

• Consider additional agreement constraint and minimize KL divergence (Ganchev et al., 2008)
Decoder for word alignment models?

• Possible, but prone to errors
  • NP-hard problem (Knight, 1999)
  • Many alternative translations with insertion/deletion
  • Spurious reordering: no distinction with local/global reordering
Overview

- Model, Training, Decoding
- Word Alignment
- Phrase-based SMT
- Evaluation
- Optimization
Phrase-based SMT

- Directly employing word-based model for decoding is not practical
- Many decisions: local/global reordering, insertion/deletion
- Use phrases to capture local reordering (at least)
Phrase extraction

- Given word alignment, contiguous phrases are extracted which do not violate alignment constraint
- Relative count-based estimation + smoothing
Decoding for phrase-based SMT

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

- Maximization by log-linear model with hidden phrase structures
- \( \Phi \): hidden variable for phrasal segmentation
- Max-derivation: searching for the best segmentation + translation
Decoding for phrase-based SMT

- left-to-right generation + bit-vector for keeping track of covered source positions
Phrase-based decoding

- NP-hard: Traveling salesman problem
Non-local features

- Example: bigram language model
- Enlarged search space
Pruning

- Beam search to limit the search space
- Multiple stack to keep hypotheses sharing the same # of covered source words
Overview

- Model, Training, Decoding
- Word Alignment
- Phrase-based SMT
- Evaluation
- Optimization
Evaluation

• How do you know translations are good or bad?

• Human judgement
  • Fluency/Adequacy, Human Translation Error Rate (H-TER), Ranking etc.

• Automatic measures: Bleu, Meteor, TER etc.
  • Uses reference translations
Evaluation: ngram precision

Well, I'd like to stay five nights beginning October twenty-fifth to thirty.

\[ p_1 = \frac{11}{15} \quad p_2 = \frac{5}{14} \quad p_3 = \frac{3}{13} \quad p_4 = \frac{2}{12} \]

- I'd like to stay there for five nights, from October twenty-fifth to the thirtieth.
- I want to stay for five nights, from October twenty-fifth to the thirtieth.
- I'd like to stay for five nights, from October twenty-fifth to the thirtieth.
- I would like to reserve a room for five nights, from October twenty-fifth to the thirtieth.
Evaluation: BLEU

\[
\exp \left( \sum_{n=1}^{N} w_n \log p_n + \min \left(1 - \frac{r}{c}, 0\right) \right)
\]

- ngram precision: weighted combination
- brevity penalty: penalize too short sentences
  - \( r = \text{reference length}, c = \text{candidate length} \)
- Both factors are computed over the whole document
Overview

• Model, Training, Decoding
• Word Alignment
• Phrase-based SMT
• Evaluation
• Optimization
Optimization: MERT

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

- Minimum Error Rate Training (MERT): directly minimize error (or max-BLEU)
- Small # of real valued features (up to 10?)
- Many local-optima, potential overfitting
MERT

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

- Generate and merge nbest list across iterations (line 3 and 4)
- Powell’s method (or coordinate descent) to perform minimization (line 5-10)
MERT: reduction to 1-dim search

- If we fix one parameter, it is one dimensional search
- Compute convex hull over a set of lines

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

• We quickly reviews basics of SMT:
  • Model, Training, Decoding
  • Word alignment
  • Phrase-based SMT
  • Evaluation
  • Optimization
SMT: Softwares

- GIZA++, gizapp, mgiza: translation model
  - mgiza: http://geek.kyloo.net/software/doku.php

- Alignment by joint training
  - Berkeley Aligner: http://code.google.com/p/berkeleyaligner/
  - PostCAT: http://www.seas.upenn.edu/~strctlrn/CAT/CAT.html

- language models

- phrase-based SMT
  - Moses: http://www.statmt.org/moses/
References


References


Tree-based SMT
Hierarchical Phrase-based SMT

X₁

X₂

X₃

X₄

X₅

言語

コミュニケーション

X₁

X₂

X₃

X₄

X₅

は

ある

言語の

コミュニケーション

コミュニケーションの

a means

communication

is

language

of
Syntax-based MT

S
  └── NP
      └── NNP
             VBP
                   └── NP
                        └── PP
                                   └── NN
                                         └── DT
                                              └── NN
                                                  └── IN
                                                      └── NP
                                                            └── NN

language is a means of communication

言語はコミュニケーションの道具である
Many variants...

<table>
<thead>
<tr>
<th>tree</th>
<th>(partial) examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>source</td>
<td>Huang et al. (2006), Liu et al. (2006), Quirk et al. (2005)</td>
</tr>
<tr>
<td>target</td>
<td>Galley et al. (2004), Shen et al. (2008)</td>
</tr>
<tr>
<td>both</td>
<td>Ding and Palmer (2005), Liu et al. (2009)</td>
</tr>
</tbody>
</table>

- formally syntactical, linguistically syntactical
- dependency structure and constituency structure
- \{tree,string\}-to-\{tree,string\}
- In this talk, we will summarize them as “tree-based MT”
Overview

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

• Tree-based SMT
  • Synchronous-CFG
  • String-to-Tree/Tree-to-String
  • Bitext parsing
Backgrounds: CFG

\[
\begin{align*}
S & \rightarrow NP \ VP \\
NP & \rightarrow NNP \\
NP & \rightarrow NP \ PP \\
NP & \rightarrow DP \ NN \\
NNP & \rightarrow language \\
VP & \rightarrow VBZ \ NP \\
VBZ & \rightarrow is \\
DT & \rightarrow a \\
\end{align*}
\]

\[
\begin{align*}
S & \rightarrow NP \ VP \\
NP & \rightarrow NNP \\
NP & \rightarrow NP \ PP \\
NP & \rightarrow DP \ NN \ language \ is \\
NNP & \rightarrow language \\
VP & \rightarrow VBZ \ NP \\
VBZ & \rightarrow is \\
DT & \rightarrow a \\
\end{align*}
\]

- parsing = intersection problem
Parsing: CKY

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

- (Bottom-up) topological order

Language is a means of communication.
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

(Shieber et al., 1995)
Packed forest

- A polynomial space encoding of exponentially many parses by sharing common sub-derivations

- Single derivation = tree

(Klein and Manning, 2001; Huang and Chiang, 2005)
Translation as parsing

- CFG to synchronous-CFG as in FST with input/output symbols
- Parsing performed over source-yield
- Translation = target-yield of a derivation

\[ VP \rightarrow \langle VP, VP, VP \rangle \]

\[ NP \rightarrow \langle NP, PP, NP \rangle \]

• CFG to synchronous-CFG as in FST with input/output symbols
• Parsing performed over source-yield
• Translation = target-yield of a derivation
Translation as tree-rewrite

- Formalized as tree transducer, tree substitution grammar, or simply, tree-rewrite system
- \{tree, string\}-to-\{tree, string\} transformation

(language is a means of communication) 

(Galley et al., 2004; Liu et al., 2006; Huang et al., 2006)
Weights and Semirings

- associate weights as in WFST
- $\otimes$: extension (multiplicative), $\oplus$: summary (additive)

\[
\begin{align*}
\text{NP} & \xrightarrow{w} \langle \text{NP}_1 \text{ PP}_2, \text{ PP}_2 \text{ NP}_1 \rangle \\
S & \\
\text{NP} & \xrightarrow{w} \text{ x}_2: \text{VP} \\
x_1: \text{NNP} & \xrightarrow{w} x_1 \text{ x}_2 \\
\end{align*}
\]

\[
\begin{align*}
\text{NP},_2,6 & : w \otimes a \otimes b \\
\text{NP},_2,4 & : a \quad \text{PP},_4,6 : b \\
\text{NP},_2,4 & : a \quad \text{PP},_4,6 : b \\
\frac{\text{NP},_2,4 : a \quad \text{PP},_4,6 : b}{\text{NP},_2,6 : w \otimes a \otimes b} : w \\
\end{align*}
\]

(Goodman, 1999)
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

\[ d(v) = (w(e_1, u_1, u_2) \otimes d(u_1) \otimes d(u_2)) \]

\[ \oplus (w(e_2, u_3, u_4) \otimes d(u_3) \otimes d(u_4)) \]

- 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
Summary

• Synchronous-CFG: context free rewrite system whose right-hand-side is paired

• Special instances:
  • Inversion Transductive Grammar (ITG) (Wu, 97)
  • Hiero Grammar (Chiang, 2007)

• \{\text{tree, string}\}-to-\{\text{tree, string}\} models
  • Recursive tree rewriting
  • Formalized as tree transducer or tree substitution grammar
Overview

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

• Tree-based SMT
  • Synchronous-CFG
  • String-to-Tree/Tree-to-String
  • Bitext parsing
Synchronous-CFG

- Derivation: single tree
- Yield: terminals covered by derivation
  - source yield = input sentence
  - target yield = translation
Synchronous-CFG: Model

\[
\text{S} \rightarrow \langle S_1 \ X_2, S_1 \ X_2 \rangle \\
\text{S} \rightarrow \langle X_1, X_1 \rangle \\
\text{X} \rightarrow \langle X_1 \ の \ X_2, X_2 \ of \ X_1 \rangle \\
\text{X} \rightarrow \langle \text{道具}, a \ \text{means} \rangle \\
\text{VP} \rightarrow \langle \text{VBZ}_1 \ \text{NP}_2, \text{VBZ}_1 \ \text{NP}_2 \rangle \\
\text{NP} \rightarrow \langle \text{NP}_1 \ \text{PP}_2, \text{PP}_2 \ \text{NP}_1 \rangle
\]

- Use only two categories, S and X (Chiang, 2007)
- Or, borrow linguistic categories from syntactic parse (Zollman and Venugopal, 2006)
Synchronous-CFG: Extraction

- From word alignment annotated data, extract phrases
- Sub-phrases treated as non-terminal
**Synchronous-CFG: Extraction**

- **Borrow syntactic categories either from source or target parse tree**

- **When no syntactic categories assigned:**
  - Try combination(+) or subtraction(/ or \) as in Combinational Category Grammar (CCG)
Synchronous-CFG: Parsing

- Translation with SCFG = monolingual parsing
- Parse the input with the source side, build projected target side in parallel
- Complexity: the same as CKY algorithm: $O(n^3)$
As in phrase decoding with non-local features (i.e. ngram), it is the same as the CKY algorithm with enlarged search space.
Cube Pruning: Basics

- Lazily enumerate top most items
- Vertices are sorted according to its score
- Pop an item from a priority queue, then expand

$w(e_1, u_1, u_4) \otimes d(u_1) \otimes d(u_4)$
$w(e_1, u_1, u_5) \otimes d(u_1) \otimes d(u_5)$
$w(e_1, u_2, u_5) \otimes d(u_2) \otimes d(u_5)$

(Chiang, 2007; Huang and Chiang, 2007)
Simultaneously process the rules sharing the same rhs and span by placing “cubes” in a priority queue.

(Chiang, 2007; Huang and Chiang, 2007)
Overview

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

• Tree-based SMT
  • Synchronous-CFG
  • String-to-Tree/Tree-to-String
  • Bitext parsing
Tree rewriting rules: each rule consists of (sub-)tree structures

Flat structure = synchronous-CFG
• We can handle various transfer rules:

• phrasal translation, non-constituent phrase, non-contiguous phrase, insertion/deletion, multi-level reordering, lexicalized reordering, long distance reordering, etc.

(Galley et al., 2004)
Rule extraction

- Compute target spans

(Galley et al., 2004)
Rule extraction

• Find admissible nodes

(Galley et al., 2004)
Rule extraction

• Extract minimum rules

(Galley et al., 2004)
Compound rules

- Tree substitution for compound rules, like phrases from a sequence of words

(Galley et al., 2006)
String-to-{string, tree} decoding

- Similar to SCFG: use flipped string side to perform CKY parsing
- After parsing, tree-reranking from forest

(Galley et al., 2004; Huang and Chiang, 2007)
Tree-to-\{string, tree\} decoding

- Recursively transform by pattern matching over tree
- After matching, forest is rescored (Huang and Chiang; 2007)

(Huang et al., 2006; Liu et al., 2006)
Overview

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

• Tree-based SMT
  • Synchronous-CFG
  • String-to-Tree/Tree-to-String

• Bitext parsing
Bitext parsing

- Bitext parsing takes $O(n^6)$ (Wu, 1997)
- For each length $n$ and $m$, for each position $i$ and $j$, for each rule $X \rightarrow \text{LHS}$, for each split point $k$ and $l$
- Fast span pruning by $O(n^3)$ (Zhang et al., 2008)
Bitext parsing: two-parse

- Parse source side (Intersect with source side)
- Extract target rules from forest (relabel category)
- Parse target side by extracted rules (Compose with target side)
- The same worst case $O(n^6)$, but fast in practice

(Dyer, 2010)
Summary

• We reviewed some backgrounds on CFG
• Tree based MT are formulated as
  • synchronous-CFG or tree-rewrite system
  • Cube pruning allows parsing with non-local features (ngrams)
Software

- Synchronous-CFG
  - Cdec: http://cdec-decoder.org
  - Jane: http://www-i6.informatik.rwth-aachen.de/jane/
  - Joshua: http://joshua.sourceforge.net
  - Moses: http://www.statmt.org/moses/
- {Tree, String}-to-{tree, string}
  - Tiburon: http://www.isi.edu/licensed-sw/tiburon/
References

References


References


• C. Dyer, “Two monolingual parses are better than one (synchronous parse),” in Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, (Los Angeles, California), pp. 263--266, Association for Computational Linguistics, June 2010.
Advanced Topics
Overview

- More data, better translation?
- Translation by many features
- Single path/derivation to lattice/forest
- Word alignment, phrases, rules
More data, better translation?

- Do we really need more data?
- Experiments on Japanese-to-English patent data
  - Language model: 11G words
  - Translation model: 108M words
Experiments: Fixed LM

- Fixed LM (11G words, 5-grams), reduced TM data (108M words)
Experiments: Fixed TM

- Fixed TM (108M words), reduced LM data (11G words)
Data handling

• Parallelization (Zhang et al., 2006; Brantz et al., 2007)
  • Split data and store in clusters
  • Efficient protocol to retrieve data

• Suffix arrays (Callison-burch and Bannard, 2005; Zhang and Vogel, 2005; Lopez, 2007)
  • raw data + index by suffix array + on-the-fly phrase/rule extraction

• Alternative solutions?
  • Randomized data structures
  • Succinct data structures
Randomized data structures

- We do not store exactly, but keep signatures (Bloom, 1970)
- Allow “false positives”
  - Not inserted, but the signature says, “exists”
- Error rate is bounded theoretically and practically
• Insert: set bits by k hash functions for m bits array
• Query: test by k hash functions
• False positives are controlled by k and m
Randomized LM

1: for \( j = 1 \ldots \) do
2: \hspace{1em} for \( i = 1 \ldots k \) do
3: \hspace{2em} if \( BF[h_{i}(\{x, j\})] = 0 \) then
4: \hspace{3em} return \( E[c(x)|qc(x) = j - 1] \)
5: \hspace{2em} end if
6: \hspace{1em} end for
7: end for

(Talbot and Osborne, 2007a, 2007b)

- Store quantized log-count: \( qc(x) = 1 + \lfloor \log_b c(x) \rfloor \)
- Returns expected count: \( E[c(x)|qc(x) = j] = \frac{b^{j-1} + b^j - 1}{2} \)
- False positives are further controlled by n-gram property:
  - If an n-gram exists, lower order (n-1)-grams exist.
  - If an n-gram exists, its count is smaller than or equal to its lower order (n-1)-grams
Randomized LM: Experiments

**French-English Europarl data**

(Talbot and Osborne, 2007a, 2007b)
Other randomized variants

• Perfect hash function based randomized storage (Talbot and Brants, 2008)
• Bloomier filter which allows dynamic insertion/deletion (Levenberg and Osborne, 2009)
Succinct data structures

• In NLP applications (including MT), models are compactly stored by trie structures (ngrams, phrase tables, grammar etc.)

• Trie structure (pointers) can be succinctly encoded by $2M + O(M)$ bits, approaching information-theoretical bounds (Jacobson, 1989):

\[
\lg \left[ \frac{1}{2M + 1} \binom{2M + 1}{M} \right] \approx 2M - O(\lg M)
\]

• An example: Level-Order Unary Degree Sequences (LOUDS) (Jacobson, 1989; Delpratt et al., 2006)
LOUDS

- Traverse in level order, left-to-right, emit 1s and 0 at each node
- $2M + 1$ bits

<table>
<thead>
<tr>
<th>node id</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit position</td>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</td>
<td>14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOUDS bit</td>
<td>101111011101100101100100110000000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

 nodenode id | 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
| bit position | 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 |
| LOUDS bit | 101111011101100101100100110000000000000 |
LOUDS: traversal

\[ \text{parent}(x) = \text{rank}_0(\text{select}_1(x + 1)) - 1 \]
\[ \text{first_child}(x) = \text{rank}_1(\text{select}_0(x + 1)) \]

- select \(1\)(x): left-most position of the \(x\)-th bits
- rank \(1\)(x): \# of bits to the left of, and including, \(x\)

<table>
<thead>
<tr>
<th>node id</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit position</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>LOUDS bit</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

97
**LOUDS: traversal**

\[
\begin{align*}
\text{parent}(x) & = \text{rank}_0(\text{select}_1(x + 1)) - 1 \\
\text{first}_\text{child}(x) & = \text{rank}_1(\text{select}_0(x + 1)) \\
\end{align*}
\]

- **parent(9):**
  \[
  \text{select}_1(9 + 1) = 12 \\
  \text{rank}_0(12) - 1 = 2 \\
  \]

- **first_\text{child}(9):**
  \[
  \text{select}_0(9 + 1) = 23 \\
  \text{rank}_1(23) = 14 \\
  \]

<table>
<thead>
<tr>
<th>node id</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit position</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>LOUDS bit</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Succinct ngram language model

- Remove root (2 bits)
- Remove the last zeros (5 bits)
- Remove unigram bits (4 + 1 bits)

\[
2N_1^N + 3 \rightarrow 2N_1^N - (N_1 + N_N)
\]

<table>
<thead>
<tr>
<th>node id</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit position</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>LOUDS bit</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(watanabe et al., 2009)
# Web-1T ngrams

<table>
<thead>
<tr>
<th></th>
<th>English</th>
<th>Chinese</th>
<th>Japanese</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>gzip size</strong></td>
<td>25G</td>
<td>25G</td>
<td>30G</td>
</tr>
<tr>
<td><strong>counts</strong></td>
<td>12.6G</td>
<td>13.2</td>
<td>9.8G</td>
</tr>
<tr>
<td><strong>quantized-lm</strong></td>
<td>13.1G</td>
<td>13.8G</td>
<td>10.7G</td>
</tr>
</tbody>
</table>

- Web 1T ngrams from Google (Chinese, English, Japanese)
Software

- Randomized LM
  - randlm: http://sourceforge.net/projects/randlm/
- (generic) succinct storage
  - tx: http://code.google.com/p/tx-trie/
  - taiju: http://code.google.com/p/taiju/
Overview

- More data, better translation?
- Translation by many features
- Single path/derivation to lattice/forest
- Word alignment, phrases, rules
Model with many features

- We want fine-grained translations
- Many binary features to represent complex decision
- MERT can handle small # of features (around 10+)
- Can we scale to millions for better translations?
Large margin training

\[
\hat{w} = \arg\min_w \frac{\lambda}{2} \|w\|^2 + \sum_{s=1}^{S} \max (l_s - w \cdot \Delta h_s)
\]

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

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

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

- Major difference to MERT is the explicit L\{1,2\} regularizer and regression term

- Very slow convergence by SMO... faster algorithms?
(Averaged) Perceptron

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: \( \hat{e} = \text{GEN}(f_s, w^{t-1}) \)

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 \( \frac{1}{N} \sum_{i=1}^{N} w^j \)

- Scales very well to very large data and large feature set

- Liang et al. (2006) reported good performance
MIRA

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

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

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

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

• line 7 of weight update is replaced by the solution of the above equation

• Similar to large margin constraints

• Experimented by: Watanabe et al. (2007); Chiang et al. (2008); Chiang et al. (2009)
Correct translations?

\[
\hat{e}\_s = \arg\max\_{e} \mathbf{w} \cdot \mathbf{h}(e, f_s) - \text{BLEU}_s(e).
\]

\[
e\_s^{*} = \arg\max\_{e} \mathbf{w} \cdot \mathbf{h}(e, f_s) + \text{BLEU}_s(e).
\]

- Problem: we cannot generate translations exactly the same as reference translations.

- Solution: select translations among nbests with “error bias” (Chiang et al., 2008; Chian et al., 2009)
MIRA: Experiments

- Consistent improvements over MERT
- Scales well to millions of features

(Chiang et al., 2009)

<table>
<thead>
<tr>
<th>System</th>
<th>Training</th>
<th>Features</th>
<th>#</th>
<th>Tune</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hiero</td>
<td>MERT</td>
<td>baseline</td>
<td>11</td>
<td>35.4</td>
<td>36.1</td>
</tr>
<tr>
<td></td>
<td>MIRA</td>
<td>syntax, distortion</td>
<td>56</td>
<td>35.9</td>
<td>36.9*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>syntax, distortion, discount</td>
<td>61</td>
<td>36.6</td>
<td>37.3**</td>
</tr>
<tr>
<td></td>
<td></td>
<td>all source-side, discount</td>
<td>10990</td>
<td>38.4</td>
<td>37.6**</td>
</tr>
<tr>
<td>Syntax</td>
<td>MERT</td>
<td>baseline</td>
<td>25</td>
<td>38.6</td>
<td>39.5</td>
</tr>
<tr>
<td></td>
<td>MIRA</td>
<td>baseline</td>
<td>25</td>
<td>38.5</td>
<td>39.8*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>overlap</td>
<td>132</td>
<td>38.7</td>
<td>39.9*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>node count</td>
<td>136</td>
<td>38.7</td>
<td>40.0**</td>
</tr>
<tr>
<td></td>
<td></td>
<td>all target-side, discount</td>
<td>283</td>
<td>39.6</td>
<td>40.6**</td>
</tr>
</tbody>
</table>

Table 5: Adding new features with MIRA significantly improves translation accuracy. Scores are case-insensitive IBM B__likelihood scores. * or ** = significantly better than MERT baseline with < 4.49 or 4.45 respectively.

Table 6: Weights learned for discount features. Negative weights indicate bonuses; positive weights indicate penalties.

We implemented the source-side context features for Hiero and the target-side syntax features for the syntax-based system, and the discount features for both. We then ran MIRA on the tuning set with 6 parallel learners for Hiero and 7 parallel learners for the syntax-based system. We chose a stopping iteration based on the B_likely score on the tuning set, and used the averaged feature weights from all iterations of all learners to decode the test set.

The results show significant improvements in both systems over already very strong MERT baselines. Adding the source-side and discount features to Hiero yields a +5.29 B_likely improvement, and adding the target-side syntax and discount features to the syntax-based system yields a +5.25 B_likely improvement. The results also show that for Hiero, the various classes of features contributed roughly equally; for the syntax-based system, we see that two of the feature classes make small contributions but time constraints unfortunately did not permit isolated testing of all feature classes.
Overview

- More data, better translation?
- Translation by many features
- Single path/derivation to lattice/forest
- Word alignment, phrases, rules
Forest approaches

• single \{tree, string\} input and single \{tree, string\} output

• As in lattice/word graph, we can compactly represent alternative derivations by forest

• Translation from forest, Extraction from forest, MBR by forest, MERT by forest
Translation from forest

- (Try) avoid errors propagated from parse tree, and make decision later

- Tree rewrite on forest, yielding larger translation forest

(Mi et al., 2008)
Translation from forest

- Faster than translating each of k-best trees
- Better translations from packed forest
Extraction from forest

- Extract more rules from forest

(Mi and Huang, 2008)
Extraction from forest

- Faster than extraction from individual trees
- Better translations from larger forest
MBR by forest

\[ \hat{e} = \arg\min_{e} \mathbb{E}_{P(e'|f)} [l(e; e')] \]
\[ = \arg\min_{e} \sum_{e'} l(e; e') P(e'|f) \]

• Instead of maximization, we reduce expected loss (MBR, Minimum Bayes Risk)

• Conventional approaches enumerate over n-best-list (Kumar and Byrne, 2004)
**MBR by linear BLEU**

\[ l(e; e') = \theta_0 |e| + \sum_{w \in N} \theta_{|w|} c_w(e) \delta_w(e') \]

\[ \hat{e} = \arg\max_{e \in G} \theta_0 |e| + \sum_w \theta_{|w|} c_w(e) p(w|G) \]

- When computing expected loss (\( = 1.0 - \text{BLEU} \)) over lattice/forest, use linearly approximated BLEU (Tromble et al., 2008, Kumar et al., 2009)
MBR by expected BLEU

\[ \text{BLEU}(e; e') = \exp \left( \min(1 - \frac{|e'|}{|e|}) + \frac{1}{4} \sum_{n=1}^{4} \log p_n(e, e') \right) \]

\[ p_n(e, e') = \frac{\sum_{w \in T, |w|=n} \min(c(e, w), c(e', w))}{\sum_{w \in T, |w|=n} c(e, w)} \]

- As an alternative to MBR, compute similarities by expected ngram statistics (DeNero et al., 2009)

- expected ngram counts for e’ are collected from hypergraph T
MERT by forest

- MERT is performed over forest, not n-best
- Hyperedge: combine lines from antecedents
- Node: Compute convex hulls for maximization

(Kumar et al., 2009)
Overview

• More data, better translation?
• Translation by many features
• Single path/derivation to lattice/forest
• Word alignment, phrases, rules
Word alignment, phrases, rules

- Better word alignment learning?
  - We learned “unsupervised” word alignment training
- What if “gold standard” exists?
- Better phrases, rules?
  - We can extract phrases/rules from word alignment annotated data
  - Can we directly induce phrases/rules?
Supervised word alignment

• IBM Models and HMM model can learn from bilingual sentences
• No control on “how word will be aligned”
• Assuming small data with word alignment annotation
• max-matching, ITG, Block-ITG, ITG+bi-parse
Max-matching alignment

\[
\begin{align*}
\max_z & \quad \sum_{j,k} s_{jk} z_{jk} \\
\text{s.t.} & \quad \sum_j z_{jk} \leq 1, \sum_k z_{jk} \leq 1, 0 \leq z_{jk} \leq 1 \\
\end{align*}
\]

\[s_{jk} = w \cdot h(e_j, f_k)\]

• Word alignment as a max-flow problem over bipartite graph (Taskar et al., 2005)
• Solved by the linear program
• Max-margin training for parameter estimation
ITG alignment

- Binary branching rules
- non-ambiguous deletion by Haghighi et al. (2009)
- Learning by EM-algorithm (Wu, 1997), or, max-margin training (Cherry and Lin, 2006)

\[
\begin{align*}
X & \rightarrow [X \ X] & X & \rightarrow \langle X_{1} \ X_{2}, X_{1} \ X_{2} \rangle \\
X & \rightarrow \langle X \ X \rangle & X & \rightarrow \langle X_{1} \ X_{2}, X_{2} \ X_{1} \rangle \\
X & \rightarrow e/f & X & \rightarrow \langle e, f \rangle 
\end{align*}
\]
ITG-alignment: Experiments

<table>
<thead>
<tr>
<th>Method</th>
<th>Prec</th>
<th>Rec</th>
<th>AER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matching</td>
<td>0.916</td>
<td>0.860</td>
<td>0.110</td>
</tr>
<tr>
<td>D-ITG</td>
<td>0.940</td>
<td>0.854</td>
<td>0.100</td>
</tr>
<tr>
<td>SD-ITG</td>
<td>0.944</td>
<td>0.878</td>
<td>0.086</td>
</tr>
</tbody>
</table>

(Cherry and Lin, 2006)

- Experiments with dependency constraint
- Evaluated by alignment error rate (AER)
- Still, it is not clear whether improved alignment implies improved BLEU
Block ITG-alignment

- Allow phrasal alignment by adding phrasal lexical rules (Haghighi et al., 2009)
Block ITG-alignment: Experiments

<table>
<thead>
<tr>
<th>Alignments</th>
<th>Translations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Prec</td>
</tr>
<tr>
<td>GIZA++</td>
<td>62</td>
</tr>
<tr>
<td>Joint HMM</td>
<td>79</td>
</tr>
<tr>
<td>Viterbi ITG</td>
<td>90</td>
</tr>
<tr>
<td>Posterior ITG</td>
<td>81</td>
</tr>
</tbody>
</table>

(Haghighi et al., 2009)

- Chinese/English translation
- Large margin-based MIRA training and MaxEnt training
- The first work to show gain by alignment improved BLEU
• ITG-alignment with syntactic parses from source/target
• Asynchronous features: no direct pairing features
• Mean field inference for approximate estimation
ITG + Bi-parsing alignment

### Table 10: Parsing Results

<table>
<thead>
<tr>
<th></th>
<th>Test Results</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>AER</td>
<td>F₁</td>
</tr>
<tr>
<td>HMM</td>
<td>86.0</td>
<td>58.4</td>
<td>30.0</td>
<td>69.5</td>
</tr>
<tr>
<td>ITG</td>
<td><strong>86.8</strong></td>
<td>73.4</td>
<td>20.2</td>
<td>79.5</td>
</tr>
<tr>
<td>Joint</td>
<td>85.5</td>
<td><strong>84.6</strong></td>
<td><strong>14.9</strong></td>
<td><strong>85.0</strong></td>
</tr>
</tbody>
</table>

### Table 3: Tune and Test LEU Results for Machine Translation

- **8.3 Machine Translation**

  - Our experiments involved tuning and testing our interpretations using various alignment tools.
  - Table 3 describes the results of train the Joshua machine translation system.
  - We further tested our alignments by using them to the current top-performing English-Chinese bilingual reranker.

- **9 Conclusion**

  - We improved results on both of these tasks, as well as parsing and alignment, demonstrating that we can improve results on both of these tasks, as well as parsing and alignment.

- **Acknowledgements**

  - We thank Dam Pauls and John DeNero for their helpful comments on an earlier draft of this paper.

- **Gain from Haghighi et al. (2009)**

  - (Burkett et al., 2010)
Direct phrase/rule induction

- We have separated word alignment and phrase/rule induction
- Can we learn directly?
Direct phrase training

• Instead of training from word alignment data, why not directly train phrases, rules?

• Many work: Marcu and Wong (2002) etc.

• Some of the problems:
  • Very expensive summation
  • EM-algorithm w/o control by prior belief: use of non-parametric Bayesian approach
### Optimization/Summation

<table>
<thead>
<tr>
<th></th>
<th>Optimization</th>
<th>Summation</th>
</tr>
</thead>
<tbody>
<tr>
<td>tractable</td>
<td>A* / Knuth / Viterbi</td>
<td>forward-backward / inside-outside</td>
</tr>
<tr>
<td>intractable</td>
<td>beam search</td>
<td>???</td>
</tr>
</tbody>
</table>

- We need summation for training parameters
- Margin-based or Loss-based learning avoid this problem
- DP-based algorithm is applicable to tractable models
- Our choice: tractable simpler (and often approximated) model or complex model w/o approximation?
Monte Carlo algorithms

- Instead of DP based summing, sampling

\[ p(Y = \{ \text{tree, alignment} \}|X = \{ \text{I parse \ldots, 私は \ldots} \}) \]
Markov Chain Monte Carlo

- Sampling by a series of small changes
Summation problem: Summary

• MCMC for intractable models
• Define your sampling operations
• Examples:
  • Phrase-based models (DeNero et al., 2008; Arun et al., 2009)
  • Synchronous-CFG (Blunsom et al., 2009)
  • string-to-tree (Cohn and Blunsom, 2009)
MCMC: efficient samplings

• Block sampling (Cohn and Blunsom, 2010):
  • Allow larger changes by simultaneously perform small changes

• Slice sampling (Blunsom and Cohn, 2010):
  • Together with block sampling, pruning parameter determined by model

• Randomized pruning (Bouchard-Coˆte´ et al., 2009):
  • Sampling over “invalid spans” instead of trees
Summary

- Promising direction by nonparametric Bayesian approaches
- Sampling methods replace DP-based training
- Alternative: Variational approaches inspired by DP-based training
Conclusion
Outlook: Progress in 20 years

• Modeling: word to phrase, tree, forest
• Search: even with complex structural modeling, we can search efficiently
• Training: large contribution from Machine Learning techniques
• Computer Science: CPU, memory, parallelization, data structure
Outlook: Future?

- More data with less structure or less data with more structures
- General translation or task-specific translation
- Your contributions!
References


• D. Talbot and T. Brants, ``Randomized language models via perfect hash functions,'' in Proceedings of ACL-08: HLT, (Columbus, Ohio), pp. 505--513, June 2008.
References


References


References


References


References


• A. Bouchard-Coˆte´, S. Petrov, and D. Klein, "Randomized pruning: Efficiently calculating expectations in large dynamic programs," in *Advances in Neural Information Processing Systems 22*.
