Improving a Statistical MT System with Automatically Learned Rewrite Patterns

Fei Xia and Michael McCord (Coling 2004)

UW Machine Translation Reading Group
Presented by Kevin Duh
Nov 3, 2004

Motivation

- Limitation of current phrase-based SMT
  - No mechanism for expressing and using linguistic phrases in reordering
  - Ordering of target words do not respect linguistic phrase boundaries
- Xia and McCord's solution:
  - Extract linguistic rewrite rules from corpora
  - Preprocess source sentences so phrase ordering is similar to that of target language
  - Perform SMT decoding with monotonic ordering constraint

Phrase-based SMT

- Current state-of-the-art SMT are phrase-based
- Use Viterbi alignment on words to extract "phrase-pairs"
  - Eg. Do word alignment in both directions, then take intersection.
- Advantage over word-based SMT
  - Memorize translation of group of words
  - Alleviate problems of translation selection, function word insertion, ordering of target words with a "phrase".

But..

- "Phrase" in Phrase-based MT is not a real linguistic phrase
- Let's call it "Clump" based MT instead from now on.
- Disadvantage of clumps:
  - No mechanism for expressing and using generalization that accounts for linguistic phrase reordering
  - Reordering of clumps does not respect linguistic phrase boundaries

Syntax-based MT

- E.g. (McCord & Bernth 1998)
- Express generalizations explicitly:
  - E.g. "Adj N" --> "N Adj"
- Rewrite rule is performed with respect to parse tree, so reordering respects linguistic phrases
- BUT…
  - Requires parsers, translation lexicon, rewrite patterns

Combined MT systems

- Och et al, 2004
  - Post-processing approach:
    - Use variety of features (from no syntax to deep syntax) to rerank N-best list from a clump-based SMT
  - But observed little improvement from syntax-based features
- Xia and McCord (this paper)
  - Pre-processing approach:
    - Use automatically learned rewrite rules to reorder source sentence into target sentence ordering, then apply clump-based SMT.

Baseline Clump SMT

- Phrase-based unigram model [Tillman & Xia (HLT 2003)]
- Maximizes probability of block sequence $b_{1:n}$
- $Pr(b_{1:n}) \approx \pi_{i=1:n}\{p(b_i)\ p(b_i\,|b_{i-1})\}$
  - $p(b_i)$ = block unigram model (joint model)
  - $p(b_i\,|b_{i-1})$ uses trigram language model (prob of first word in target b_i conditioned on final two words in b_{i-1}
- Simple set of model parameters; no distortion prob.

Baseline Clump SMT: Training

- Step 1: Word alignment

France is the first western country

la france est le premier pays occidental

- Step 2: Extract clump pairs

France => france, France => la france
is => est, first => premier
is the first => est le premier
first western country => premier pays occidental
western country => pays occidental

Baseline Clump SMT: Decoding

- Translate: “He is the first international student”
- Relevant parts of clump dictionary

is => est, first => premier, is the first => est le premier
he => il, student => e’tudiant, international => international

- One possible segmentation:
  - [He] [is the first] [international] [student]
- Possible translations:

il | est le premier | international | e’tudiant
est le premier | il | international | e’tudiant
il | international | est le premier | e’tudiant
il | est le premier | e’tudiant | international

System Overview

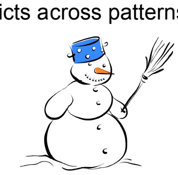
- At Training Time:
  - (T1) Learn rewrite patterns:
    - 1. Parse sentences, 2. Align phrases, 3. Extract patterns
  - (T2) Reorder source sentence using rewrite patterns
  - (T3) Train clump-based MT to get clump dictionary
- At Decoding Time:
  - (D1) Reorder test sentences with rewrite patterns
  - (D2) Translate reordered sentence in *monotonic* order
- We’ll focus on (T1) and (T2) hereafter.
- Note: need parser source sentences. Parser for target sentences is optional.

Definition of rewrite patterns

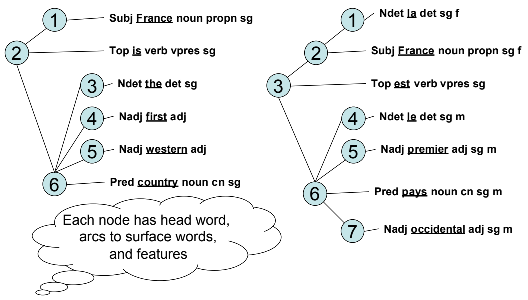
- Rewrite pattern is a quintuple:
  - (SourceRule, TargetRule, SourceHeadPosition, TargetHeadPosition, ChildAlignment)
- Rule: $l(X) \rightarrow l(X_1) \ldots l(X_m)$
  - $l(X)$ is label of node X in parse tree
  - $\{X_i\}$ must include head child of X
- ChildAlignment:
  - injective correspondence between source $\{X_i\}$ and target $\{Y_j\}$
- Simplification:
  - (NP→Adj N) => (NP→N Adj) becomes Adj N => N Adj
- Lexicalized rule:
  - Adj(good) N => Adj(bon) N

Learning Rewrite Patterns: a Four-Steps Procedure

1. Parse input sentence (Slot grammar)
2. Align phrases (based on word-alignments)
3. Extract rewrite patterns using (1) and (2) results
4. Organize rewrite patterns into an hierarchy and resolve conflicts across patterns



Parsing sentences with Slot Grammar



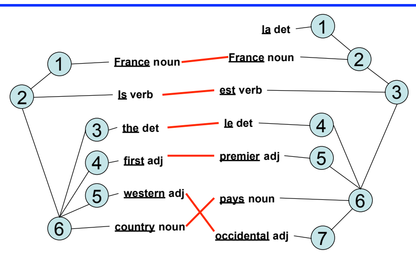
Aligning Phrases

1. Align source and target words using word aligner
2. For each source phrase S and target phrase T, calculate:

$$Score(S,T) = \frac{\#Links(S,T)}{Span(S)+Span(T)}$$

   - #Links(S,T) = total number of words linked between S and T
   - Span(X) = number of words in phrase X
3. Align S to T with the highest Score(S,T)

Aligning Phrases



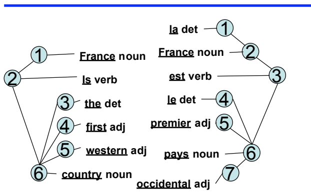
Pop quiz: What aligns best to phrase 6 in English?

Extracting Rewrite Patterns

Given a parse tree pair and a phrase alignment, extract all rewrite patterns $(X_1 \ldots X_m) \Rightarrow (Y_1 \ldots Y_n)$ that satisfies:

1. X_i are siblings and relative ordering in $(X_1 \ldots X_m)$ is the same as ordering in tree
   - => forces phrases to respect linguistic boundaries defined by the tree
2. Parent node X aligns to parent node Y
   - => or else $(X_1 \ldots X_m) \Rightarrow (Y_1 \ldots Y_n)$ aren't even phrase pairs
3. $\{X_i\}$ and $\{Y_j\}$ both contain head child, and head children must be aligned
   - => rules out un-linguistic phrases
4. Any aligned child pair is either both lexicalized or both unlexicalized
   - => allows for both specific and general rules

Extracting Rewrite Patterns



Phrase alignment:
1 => 2
2 => 3
3 => 4
4 => 5
5 => 7
6 => 6

Pop quiz:
Adj N => ???
Adj(western) N => ???
Adj N(country) => ???

Organizing Rewrite Patterns

- The pattern extraction step produces:
  - Conflicting rules: (Adj N => Adj N) vs (Adj N => N Adj)
  - Many, many patterns (due to lexicalized patterns)
- Patterns need to be organized and filtered before they can be useful
- Main ideas for organization:
  - Organize patterns by source rule
    - Because they are ultimately applied to source trees
  - Order patterns by "specificity".
    - E.g. (Adj(first) N) is more specific than (Adj N)
  - Conflicting patterns are resolved by count statistics

Algorithm for Organizing Rewrite Patterns

(Stage A) Organize patterns into a hierarchy:

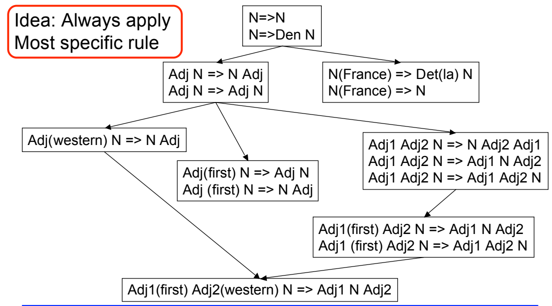
1. Patterns with the same source rule are grouped in the same group
2. Inside each group, order patterns by counts
3. For each group pair (A,B), add a link A->B iff source rule of B is more specific than A, and there is no other group between A and B

- The result is a network of rule groups

(Stage B) Filter groups to reduce hierarchy:

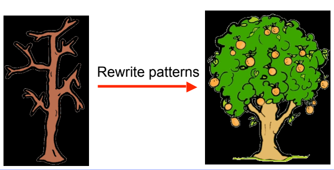
- delete a group if it is too similar to parent groups

Hierarchy of Pattern Groups



Finally…. Applying Rewrite Patterns

- Greedy algorithm:
  - Given parse tree T, iteratively apply pattern to nodes in T
  - The pattern applied is the most specific pattern possible
  - Traversal order is irrelevant since reordering will only change order of children



Experiments

- TrainSet:
  - English-French Canadian Hansard corpus
  - Extracted 2.9M patterns
  - 56k patterns after organizing/filtering
    - 1042 patterns are unlexicalized
  - Each source parse tree triggers 1.4 patterns on average
    - Common patterns: reordering of noun and its modifiers
- TestSet1:
  - 3971 Hansard sentences (not in TrainSet)
  - Ave sentence length: 21.7 words
- TestSet2:
  - TestSet2: 500 sentences from various news articles
  - Ave sentence length: 28.8 words

Results & Observations
(Refer to Fig 6 & Fig 7 in paper)

- Compare baseline and new system BLEU scores
  - Results for both TestSet1 and TestSet2.
  - Plot BLEU score against varying maximal clump length
  - Note: BLEU scores calculated from only one reference
- RESULT 1: Clump-based systems benefit from memorizing n-grams but performance saturates as n increases
  - This is because there are fewer high order n-grams that appear in both TrainSet and TestSet

Results & Observations
(Refer to Fig 6 & Fig 7 in paper)

- RESULT 2: TestSet1 curve saturates at n=4, but TestSet2 curve saturates at n=6.
  - Difference of saturation point indicates degree of similarity to TrainSet
- RESULT 3: For TestSet2, reordering is better than baseline regardless of n, but for TestSet1 this is only true for n<4
  - Together with RESULT2, this implies main benefit of reordering is for ***unseen*** source word sequences

Non-Monotonic Decoding Experiment

- Approach in Fig 6&7 is to first reorder source phrases, then translate in ***monotonic*** order
- To test effect of reordering at *target* side, allow ***non-monotonic*** reordering at decoder
  - Some form of restricted permutation was used
- BLEU scores with one reference:

	Non-Monotonic	Monotonic (Fig 6,7)
Baseline	0.187	0.196
Reordering system	0.185	0.215

Conclusion and Future Directions

- Addressed 2 limitations of clump-based SMT
- Proposed:
  - Automatic method for extracting rewrite patterns based on parse tree and phrase alignments
  - Applying rewrite patterns to source tree, then decode monotonically
- Future directions:
  - Try on language pairs with more word order difference
  - Study how parsing accuracy affects reordering and MT results
  - Use rewrite patterns directly in decoders

Discussions