Learning Non-Isomorphic Tree Mappings for Machine Translation

Jason Eisner  - Johns Hopkins Univ.

Syntax-Based Machine Translation

- Previous work assumes essentially isomorphic trees
- But trees are not isomorphic!
  - Discrepancies between the languages
  - Free translation in the training data

Synchronous Tree Substitution Grammar

Two training trees, showing a free translation from French to English.

"beaucoup d’enfants donnent un baiser à Sam" ⇒ "kids kiss Sam quite often"

Synchronous Tree Substitution Grammar

Two training trees, showing a free translation from French to English.
A possible alignment is shown in orange.

"beaucoup d’enfants donnent un baiser à Sam" ⇒ "kids kiss Sam quite often"

Synchronous Tree Substitution Grammar

Two training trees, showing a free translation from French to English.
A much worse alignment ...

"beaucoup d’enfants donnent un baiser à Sam" ⇒ "kids kiss Sam quite often"
"beaucoup d’enfants donnent un baiser à Sam" → "kids kiss Sam quite often"
Grammar = Set of Elementary Trees

Form of model of big tree pairs

Joint model $P_{\theta}(T1,T2)$.
Wise to use noisy-channel form: $P(T1 | T2) \times P(T2)$.
But any joint model will do.

Inside Probabilities

Probability model similar to PCFG

Probability of generating training trees $T1$, $T2$ with alignment $A$

$P(T1, T2, A) = \prod p(t1,t2,a \mid n)$

is given by a maximum entropy model

Maxent model of little tree pairs

$p(\text{wrongly} \leftrightarrow \text{misinform?})$
$\beta(\text{report} \leftrightarrow \text{misinform?})$
$\beta(\text{verb incorporates adverb child?})$
$\beta(\text{verb incorporates child 1 of 3?})$
$\beta(\text{children 2, 3 switch positions?})$
$\beta(\text{common tree sizes & shapes?})$
$\beta(\text{... etc. ...})$

Inside Probabilities

only $O(n^2)$
P(T1, T2, A) = \prod p(t_1,t_2,a | n)

- Alignment: find A to max P(T1,T2,A)
- Decoding: find T2, A to max P(T1,T2,A)
- Training: find \( \theta \) to max \( \sum_{A} P(T1,T2,A) \)

Do everything on little trees instead!
- Only need to train & decode a model of \( p(\theta)(t_1,t_2,a) \)
- But not sure how to break up big tree correctly
  - So try all possible little trees & all ways of combining them, by dynamic prog.

Alignment Pseudocode

For each node \( c_1 \) of \( T_1 \) (bottom-up)
  - for each possible little tree \( t_1 \) rooted at \( c_1 \)
  - for each node \( c_2 \) of \( T_2 \) (bottom-up)
    - for each possible little tree \( t_2 \) rooted at \( c_2 \)
      - for each matching \( a \) between frontier nodes of \( t_1 \) and \( t_2 \)
        - \( p = p(t_1,t_2,a) \)
      - for each pair \( (d_1,d_2) \) of frontier nodes matched by \( a \)
        - \( p = p \beta \beta \beta \beta (d_1,d_2) \) // inside probability of kids
      - \( \beta \beta \beta \beta (c_1,c_2) = \beta \beta \beta \beta (c_1,c_2) + p \) // our inside probability

Nonterminal states are used in practice but not shown here
For EM training, also find outside probabilities

An MT Architecture

- Viterbi alignment yields output \( T_2 \)
- Dynamic programming engine
  - Trainer: scores all alignments of two big trees \( T_1 \), \( T_2 \)
  - Decoder: scores all alignments between a big tree \( T_1 \) & a forest of big trees \( T_2 \)
  - For each possible \( t_1 \),\( \theta \), \( (t_1,t_2,a) \)
    - For each proposed \( t_1 \),\( (t_1,t_2,a) \)
      - For each possible \( t_1 \),\( (t_1,t_2,a) \)
        - For each proposed \( t_1 \),\( (t_1,t_2,a) \)
          - Update parameters \( \theta \)
          - Score little tree pair
          - Propose translations \( t_2 \) of little tree \( t_1 \)
    - Probability Model \( p_\theta(t_1,t_2,a) \) of Little Trees

Related Work

- Synchronous grammars (Shieber & Schabes 1990)
  - Statistical work has allowed only 1:1 (isomorphic trees)
  - Stochastic inversion transduction grammars (Wu 1995)
  - Head transducer grammars (Alishawi et al. 2000)
- Statistical tree translation
  - Noisy channel model (Yamada & Knight 2000)
  - Infers tree: trains on (string, tree) pair, not (tree, tree) pair
  - But again, allows only 1:1, plus 1:0 at leaves
- Data-oriented Translation (Petitjean 2000)
  - Synchronous DOP model trained on already aligned trees
  - Statistical tree generation
    - Similar to our decoding: construct forest of appropriate trees, pick by highest prob
    - Dynamic prog. search in packed forest (Langkilde 2000)
    - Stack decoder (Ratnaparkhi 2000)

What Is New Here?

- Learning full elementary tree pairs, not rule pairs or subcat pairs
  - Previous statistical formalisms have basically assumed isomorphic trees
- Maximum-entropy modeling of elementary tree pairs
- New, flexible formalization of synchronous Tree Subst. Grammar
  - Allows either dependency trees or phrase-structure trees
  - “Empty” trees permit insertion and deletion during translation
  - Concrete enough for implementation (cf. informal previous descriptions)
  - TSG is more powerful than CFG for modeling trees, but faster than TAG
- Observation that dynamic programming is surprisingly fast
  - Find all possible decompositions into aligned elementary tree pairs
  - \( O(n^r) \) if both input trees are fully known and elem. tree size is bounded

Status & Thanks

- Developed and implemented during JHU CLSP summer workshop 2002 (funded by NSF)
- Other team members: Jan Hajic, Bonnie Dorr, Dan Gildea, Gerald Penn, Drago Radev, Owen Rambow, and students: Martin Cmejrek, Yuan Ding, Terry Xoo, Kristen Parton
- Also being used for other kinds of tree mappings:
  - between deep structure and surface structure, or semantics and syntax
  - between original text and summarized/paraphrased/plagiarized version
- Results forthcoming (that’s why I didn’t submit a full paper☺)
Summary

- Most MT systems work on strings
- We want to translate trees – want to respect syntactic structure
- But don’t assume that translated trees are structurally isomorphic!

- TSG formalism: Translation locally replaces tree structure and content.
- Parameters: Probabilities of local substitutions (use maxent model)
- Algorithm: Dynamic programming (local substitutions can’t overlap)

- EM training on <English tree, Czech tree> pairs can be fast:
  - Align O(n) tree nodes with O(n) tree nodes, respecting subconstituency
  - Dynamic programming – find all alignments and retrain using EM
  - Faster than aligning O(n) words with O(n) words
  - If correct training tree is unknown, a well-pruned parse forest still has O(n) nodes