Structure and Complexity of Grammar-Based Machine Translation

Giorgio Satta
University of Padua, Italy

New York, June 9th, 2006

1. Introduction
2. Synchronous context-free grammars
   - Definitions
   - Computational problems
3. Intersection problem
   - SCFG projection
   - SCFG Intersection
   - Algorithms
4. Computational complexity
   - Upper bounds
   - Hardness results
   - Lower bounds

Introduction

- State of the art machine translation systems are based on mathematical **translation models**, which account for all the elementary operations that rule the translation process
- Translation models are usually enriched with statistical parameters to drive the search
- Translation models are also exploited in word/phrase alignment, multilingual document retrieval, automatic dictionary construction, bilingual corpora annotation, etc.

Introduction (cont'd)

- Early translation models based on finite-state machinery :
  - IBM model, word to word [Brown et al. 1993]
  - Phrase-based [Och et al. 1999, Och and Ney 2002]
- Finite state techniques cannot easily model translations between languages with strong differences in word ordering

Introduction (cont'd)

- Recent shift towards more powerful hierarchical translation models :
  - Inversion Transduction Grammars [Wu 1997]
  - Head Transducer Grammars [Alshawi et al. 2000]
  - Tree-to-string models [Yamada and Knight 2001], [Galley et al. 2004]
  - Loosely tree-based model [Gildea 2003]
  - Multi-Text Grammars [Melamed 2003]
  - Hierarchical phrase-based models [Chiang 2005]

Introduction (cont'd)

- Most of the translation models above can be abstractly viewed as **synchronous context-free grammars**
- Synchronous context-free grammars are rooted in the theory of compilers, where they are called **syntax-directed translation schemata** (SDTS) [Lewis and Stearns 1968], [Aho and Ullman 1969]

Synchronous context-free grammars

- A **synchronous context-free grammar** (SCFG) is based on three components :
  - Context free grammar (CFG) for source language
  - CFG for target language
  - **Pairing relation** (bijection) on the productions of the two grammars and their nonterminals
- Each rule pair called **synchronous production**
- Pairing relation between nonterminals represented by superscript integers called **indices**

Example

Fragment SCFG (English to Japanese, [Yamada and Knight 2001])

$$
\begin{array}{lll}
s_1 : & [\mathrm{VB} \rightarrow \mathrm{PRP}^{(1)}\ \mathrm{VB1}^{(2)}\ \mathrm{VB2}^{(3)}, & \mathrm{VB} \rightarrow \mathrm{PRP}^{(1)}\ \mathrm{VB2}^{(3)}\ \mathrm{VB1}^{(2)}] \\
s_2 : & [\mathrm{VB2} \rightarrow \mathrm{VB}^{(1)}\ \mathrm{TO}^{(2)}, & \mathrm{VB2} \rightarrow \mathrm{TO}^{(2)}\ \mathrm{VB}^{(1)}\ \mathrm{ga}] \\
s_3 : & [\mathrm{TO} \rightarrow \mathrm{TO}^{(1)}\ \mathrm{NN}^{(2)}, & \mathrm{TO} \rightarrow \mathrm{NN}^{(2)}\ \mathrm{TO}^{(1)}] \\
s_4 : & [\mathrm{PRP} \rightarrow \mathrm{he}, & \mathrm{PRP} \rightarrow \mathrm{kare\ ha}] \\
s_5 : & [\mathrm{VB1} \rightarrow \mathrm{adores}, & \mathrm{VB1} \rightarrow \mathrm{daisuki\ desu}] \\
s_6 : & [\mathrm{VB} \rightarrow \mathrm{listening}, & \mathrm{VB} \rightarrow \mathrm{kiku\ no}] \\
s_7 : & [\mathrm{TO} \rightarrow \mathrm{to}, & \mathrm{TO} \rightarrow \mathrm{wo}] \\
s_8 : & [\mathrm{NN} \rightarrow \mathrm{music}, & \mathrm{NN} \rightarrow \mathrm{ongaku}]
\end{array}
$$

Derivations

- A SCFG generates pairs of strings/trees, representing the desired translation
- The **derive** relation applies a synchronous production to simultaneously rewrite two paired nonterminals (nonterminals with same index)
- Pairing relation must be updated after each application of a synchronous production

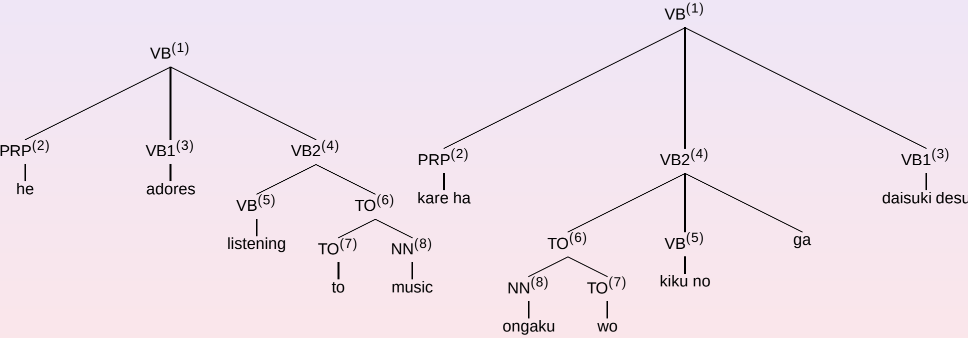
Example (cont'd)

Fragment derivation:

$$
\begin{aligned}
& [\mathrm{VB}^{(1)},\ \ \mathrm{VB}^{(1)}] \\
& \Rightarrow_G^{s_1} \ [\mathrm{PRP}^{(2)}\ \mathrm{VB1}^{(3)}\ \mathrm{VB2}^{(4)},\ \ \mathrm{PRP}^{(2)}\ \mathrm{VB2}^{(4)}\ \mathrm{VB1}^{(3)}] \\
& \Rightarrow_G^{s_4} \ [\text{he } \mathrm{VB1}^{(3)}\ \mathrm{VB2}^{(4)},\ \ \text{kare ha } \mathrm{VB2}^{(4)}\ \mathrm{VB1}^{(3)}] \\
& \Rightarrow_G^{s_5} \ [\text{he adores } \mathrm{VB2}^{(4)},\ \ \text{kare ha } \mathrm{VB2}^{(4)} \text{ daisuki desu}] \\
& \Rightarrow_G^{s_2} \ [\text{he adores } \mathrm{VB}^{(5)}\ \mathrm{TO}^{(6)},\ \ \text{kare ha } \mathrm{TO}^{(6)}\ \mathrm{VB}^{(5)} \text{ ga daisuki desu}]
\end{aligned}
$$

Example (cont'd)

Parse trees :



Translation

- Let G be a SCFG and w a string
- **Translation relation** : Set of all string pairs generated by G

$$T(G) \quad = \quad \{[u, v] \mid [S^{(1)}, S^{(1)}] \Rightarrow_G^* [u, v]\}$$

- **Image** of w : Set of strings that are translations of w

$$T(w, G) \quad = \quad \{v \mid [w, v] \in T(G)\}$$

Probabilistic SCFGs

- In a **Probabilistic SCFG**, each synchronous production associated with a probability

$$p_G([A_1 \rightarrow \alpha_1,\ A_2 \rightarrow \alpha_2])$$

- Normalization conditions for each pair $[A_1, A_2]$

$$\sum_{\alpha_1,\alpha_2} p_G([A_1 \rightarrow \alpha_1,\ A_2 \rightarrow \alpha_2]) \ = \ 1$$

PSCFGs (cont'd)

- In PSCFG we can define several joint distributions (t_i trees, w_i strings, y = yield)

$$
\begin{aligned}
p_G([t_1,\ t_2] &= \prod_{i=1}^{n} p_G(s_i) \\
p_G([w_1,\ w_2]) &= \sum_{y([t_1,t_2])=[w_1,w_2]} p_G([t_1,t_2]) \\
p_G([w_1,\ t_2]) &= \sum_{y(t_1)=w_1} p_G([t_1,t_2]) \\
&\vdots
\end{aligned}
$$

Computational Problems

- **Translation problem** : given SCFG G and string w, compute **parse forest** for strings in $T(w, G)$
- Size of parse forest for $T(w, G)$ can be a double exponential function in the size of w
- Highly compressed representation of parse forest is needed; we consider context-free grammars [Lang 1994] or, equivalently, hyper-graphs [Klein and Manning 2001]

Computational Problems (cont'd)

- **Recognition/Parsing problem** : given SCFG G and string pair $[u, v]$
  - decide whether $[u, v] \in T(G)$
  - construct parse forest for all derivations of $[u, v]$ by G
- The parsing problem is used in word/phrase alignment applications, bilingual dictionary construction, parallel corpora annotations, etc.

Computational Problems (cont'd)

- We introduce a new problem called the **intersection problem**; this generalizes the translation and the recognition/parsing problems, and several others
- We provide an abstract framework for the solution of the intersection problem
- Many of the (superficially different) translation and parsing algorithms proposed in the literature can be viewed as special cases of the above framework
- Similar attempts to define abstract frameworks for translation algorithms in [Bertsch and Nederhof 2001] and [Melamed and Wang 2005]

SCFG Projection

- We can project SCFG G into its **left and right grammar** components

$$\text{proj}(G, 1), \qquad \text{proj}(G, 2)$$

which are both CFGs

- We can similarly project the translation $T(G)$ into its **left and right language** components ($i = 1, 2$)

$$\text{proj}(T(G), i) \quad = \quad \{w_i \mid [w_1,\ w_2] \in T(G)\}$$

SCFG Projection (cont'd)

- In general the left grammar and the left language are not equivalent

$$L(\text{proj}(G, 1)) \quad \neq \quad \text{proj}(T(G), 1)$$

  (similarly for right case)
- This is because in synchronous derivations the left and right grammars interact; this is called mutual **controlled rewriting**

SCFG auto-projection

- We can efficiently construct the left and right **auto-projection** of SCFG G

$$\text{auto-proj}(G, 1), \qquad \text{auto-proj}(G, 2)$$

- The left auto-projection grammar and the left language are equivalent (similarly for right case)
- $\text{auto-proj}(G, 1)$ and $\text{auto-proj}(G, 2)$ are CFGs; this proves the **weak language preservation property** [Rambow and Satta 1996]

Intersection construction

- Let M_1, M_2 be Finite Automata (FAs); define the **Cartesian product**

$$L(M_1) \times L(M_2) \quad = \quad \{[u,\ v] \mid u \in L(M_1),\ v \in L(M_2)\}.$$

- Given SCFG G and FAs M_1, M_2, the **intersection construction** provides a new SCFG $G_\cap$ such that

$$T(G_\cap) \quad = \quad T(G) \cap (L(M_1) \times L(M_2))$$

- Parse trees are also preserved (modulo node relabeling)
- $G_\cap$ is called the **intersection** SCFG

Intersection construction (cont'd)

- $G_\cap$ has nonterminals of the form

$$(q_1, A, q_2)$$

for q_1, q_2 states of the source FAs and A a nonterminal of the source SCFG

- $G_\cap$ has productions of the form

$$\begin{array}{rcl}[(q_{10}, A_{10}, q_{1r}) & \rightarrow & (q_{10}, A_{11}, q_{11})^{(t_1)} \cdots (q_{1r-1}, A_{1r}, q_{1r})^{(t_r)}, \\ (q_{20}, A_{20}, q_{2r}) & \rightarrow & (q_{20}, A_{21}, q_{21})^{(t_{\pi(1)})} \cdots (q_{2r-1}, A_{2r}, q_{2r})^{(t_{\pi(r)})}]\end{array}$$

Translation algorithm

- Input: SCFG G, string w
- Algorithm:
  - construct M_1 such that $L(M_1) = \{w\}$
  - construct M_2 such that $L(M_2) = V_T^*$
  - construct $G_\cap$ by intersection of G with M_1 and M_2
  - output parse forest (CFG) auto-proj$(G_\cap, 2)$

Parsing algorithm

- Input: SCFG G, strings u, v
- Algorithm:
  - construct M_1 such that $L(M_1) = \{u\}$
  - construct M_2 such that $L(M_2) = \{v\}$
  - construct $G_\cap$ by intersection of G with M_1 and M_2
  - output parse forests (CFG) auto-proj$(G_\cap, 1)$, auto-proj$(G_\cap, 2)$ and synchronous parse forest (SCFG) $G_\cap$

Computational analysis

- Parameters:
  - SCFG G with maximum right-hand side length r, called **rank**
  - FA M_1 with states Q_1 and transitions δ_1
  - FA M_2 with states Q_2 and transitions δ_2
- Auto-projection can be constructed in time $\mathcal{O}(|G|)$
- In the worst case, construction of intersection grammar takes time

$$\Theta(|G| \cdot (|Q_1|^{r+1} + |\delta_1|) \cdot (|Q_2|^{r+1} + |\delta_2|))$$

Applications

- One of the very first translation algorithms has been proposed in [Wu and Wong 1998] for Stochastic Inversion Transduction Grammars (SITG)
- Translates an English sentence w into Chinese, using a filtering 2-gram language model for target language
- Algorithm runs in time $\mathcal{O}(|w|^7)$ (grammar size ignored here)
- Improved to $\mathcal{O}(|w|^6)$ in [Huang et al. 2005]

Application (cont'd)

- We can provide a very simple account of previous upper bound within our framework
  - SITG have rank $r = 2$
  - M_1 encodes w in $|w| + 1$ states
  - M_2 encodes Chinese 2-gram model in $\mathcal{O}(|w|)$ states; this is restricted to Chinese words that are image of English words in w
- Intersection algorithm then runs in time

$$\mathcal{O}(|Q_1|^{r+1} \cdot |Q_2|^{r+1}) \quad = \quad \mathcal{O}(|w|^6)$$

Application (cont'd)

- We can provide a similar polynomial time upper bound for Head Transducer Grammars [Alshawi et al. 2000]
- Polynomial time also holds if
  - SCFG is fixed; or else
  - there is a constant upper bound on the rank of the SCFG
- Otherwise, intersection construction runs in **exponential** time in the size of the input

Rank

- **Result**: SCFGs do not admit canonical forms with bounded rank [Aho and Ullman 1969] (contrast with Chomsky normal form for CFGs)
- Higher rank (flat structure) used when language pair does not satisfy **direct correspondence assumption** [Hwa et al. 2002]
- Question : Is constant upper bound on rank a plausible hypothesis for natural language translation?
  - If you need unbounded rank, your translation relation may be out of the reach of CFG analysis (scrambling, etc.)

Rank (cont'd)

- Synchronous productions that cannot be reduced in rank implement so-called **simple** permutations
- Percentage of the $r!$ permutations that are simple approaches e^{-2} [Albert et al. 2003]
- How many simple permutations are observed in real data?
- **Result**: One can decompose a rank r synchronous production into smallest rank components in time $\mathcal{O}(|r|)$ [Gildea et al. 2006]
- Above algorithm can also be used to decide whether a permutation is simple

Parsing

- **Result**: Parsing problem for SCFGs is NP-hard [Satta and Peserico 2005]
- Proof: Reduction from 3SAT; complexity comes from complex permutations
- Result transfers to translation models in [Yamada and Knight 2001], [Gildea 2003], [Melamed 2003]

Translation

- **String-to-tree** (1-best) translation problem :
  - Input a probabilistic SCFG G and a string w
  - Output the parse tree with highest probability that translates w

$$\underset{t}{argmax}\ p_G([w,\ t])$$

- **Result**: String-to-tree problem is NP-hard [Satta and Peserico 2005]
- Proof: Reduction from the consensus problem [Casacuberta and de la Higuera 2000]; complexity comes from hidden layer of source parse trees

Translation (cont'd

- String-to-tree problem remains hard even in case of constant upper bound on rank of SCFG
- Becomes polynomial time if paired nonterminals are always equal
  - Algorithm: Intersection construction + Viterbi search on right auto-projection
- Becomes undecidable if infinite ambiguity is allowed, even for a fixed SCFG !!

Parsing

- Parsing problem for SCFGs usually solved through tabular methods (chart parsing)
- if we parse left-to-right on the source sentence, we end up with **discontinuous constituents** on the target sentence
- Discontinuous constituents (multiple edges) increase the time complexity of the parser
- Are there better strategies for tabular methods?

Parsing (cont'd)

- In the worst case, tabular methods require time

$$\Theta(|G| \cdot |w|^{k(G)})$$

- We know that, unless P $=$ NP, $k(G)$ cannot be a constant
- **Result**: In the worst case, standard tabular methods for the SCFG parsing problem require an amount of time $\Omega(|G|\, n^{c \cdot \sqrt{r}})$, with r the rank of G and c some constant [Satta and Peserico 2005]
- Proof: combinatorial argument

Conclusions

- All hardness and lower bound results exploit constructions that are quite artificial
- If unbounded rank is needed, then the translation is probably out of the reach of CFG analysis
- Efficient algorithms exist for reducing rank to a minimum (expected low)
- Intersection construction extends to
  - specialized and efficient parsing strategies
  - estimation algorithm based on frequency count of synchronous productions

A. V. Aho and J. D. Ullman.
1969.
Syntax directed translations and the pushdown assembler.
Journal of Computer and System Sciences, 3(1):37–56.

M. H. Albert, M. D. Atkinson, and M. Klazar.
2003.
The enumeration of simple permutations.
Journal of Integer Sequences, 6(03.4.4):18 pages.

Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas.
2000.
Learning dependency translation models as collections of finite state head transducers.
Computational Linguistics, 26(1):45–60, March.

E. Bertsch and M.-J. Nederhof.

2001.
On the complexity of some extensions of RCG parsing.
In *Proceedings of the Seventh International Workshop on Parsing Technologies*, pages 66–77, Beijing, China, October.

Peter F. Brown, Stephen A. Della Pietra, Vincent Della J. Pietra, and Robert L. Mercer.
1993.
The mathematics of statistical machine translation: Parameter estimation.
Computational Linguistics, 19(2):263–312, June.

F. Casacuberta and C. de la Higuera.
2000.
Computational complexity of problems on probabilistic grammars and transducers.

In L. Oliveira, editor, *Grammatical Inference: Algorithms and Applications; 5th International Colloquium, ICGI 2000*, pages 15–24. Springer.

D. Chiang.
2005.
A hierarchical phrase-based model for statistical machine translation.
In *Proc. of the 43rd ACL*, pages 263–270.

Daniel Gildea.
2003.
Loosely tree-based alignment for machine translation.
In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Sapporo, Japan, July.

R. Hwa, P. Resnik, A. Weinberg, and O. Kolak.

2002.
Evaluating translational correspondence using annotation projection.
In *ACL-02*.

D. Klein and C.D. Manning.
2001.
Parsing and hypergraphs.
In *Proceedings of the Seventh International Workshop on Parsing Technologies*, pages 123–134, Beijing, China, October.

S. Kumar and W. Byrne.
2003.
A weighted finite state transducer implementation of the alignment template model for statistical machine translation.

In *Proceedings of HLT-NAACL*.

B. Lang.
1994.
Recognition can be harder than parsing.
Computational Intelligence, 10(4):486–494.

P. M. Lewis and R. E. Stearns.
1968.
Syntax-directed transduction.
Journal of the Association for Computing Machinery, 15(3):465–488.

D. Melamed and W. Wang.
2005.
Proteus project working paper #2: Statistical machine translation by generalized parsingy.

Proteus Project technical report #05-001. Available from http://nlp.cs.nyu.edu/pubs/.

I. Dan Melamed.
2003.
Multitext grammars and synchronous parsers.
In *Proceedings of the Human Language Technology Conference and the North American Association for Computational Linguistics (HLT-NAACL)*, pages 158–165, Edmonton, Canada.

Franz Josef Och and Hermann Ney.
2002.
Discriminative training and maximum entropy models for statistical machine translation.

In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, July.

Franz Josef Och, Christoph Tillmann, and Hermann Ney.
1999.
Improved alignment models for statistical machine translation.
In *Proceedings of the 4nd Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 20–28, College Park, Maryland.

O. Rambow and G. Satta.
1996.
Synchronous models of language.

In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, Santa Cruz, USA.

G. Satta and E. Peserico.
2005.
Some computational complexity results for synchronous context-free grammars.
In *Proc. of the 2005 Conference on Empirical Methods in Natural Language Processing*, Vancouver, Canada.

Dekai Wu and Hongsing Wong.
1998.
Machine translation with a stochastic grammatical channel.
In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*, Montreal, Canada, July.

Dekai Wu.
1997.
Stochastic inversion transduction grammars and bilingual parsing of parallel corpora.
Computational Linguistics, 23(3):377–404, September.

Kenji Yamada and Kevin Knight.
2001.
A syntax-based statistical translation model.
In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 531–538, Toulouse, July.