Efficient Algorithms for Richer Formalisms: Parsing and Machine Translation

Liang Huang
Department of Computer and Information Science
University of Pennsylvania
lhuang3@cis.upenn.edu

My PhD research has been on the algorithmic and formal aspects of computational linguistics, esp. in the areas of parsing and machine translation. I am interested in developing efficient algorithms for formalisms with rich expressive power, so that we can have a better modeling of human languages without sacrificing efficiency. In doing so, I hope to help integrating more linguistic and structural knowledge with modern statistical techniques, and in particular, for syntax-based machine translation (MT) systems.

Among other projects, I have been working on k-best parsing, synchronous binarization, and syntax-directed translation.

1 k-best Parsing and Hypergraphs

NLP systems are often cascades of several modules, e.g., part-of-speech tagging, then syntactic parsing, and finally semantic interpretation. It is often the case that the 1-best output from one module is *not* always optimal for the next module. So one might want to postpone some disambiguation by propagating k-best lists (instead of 1-best solutions) to subsequent phases, as in joint parsing and semantic role-labeling (Gildea and Jurafsky, 2002). This is also true for *reranking* and *discriminative training*, where the k-best list of candidates serves as an approximation of the full set (Collins, 2000; Och, 2003; McDonald et al., 2005). In this way we can optimize some complicated objective function on the k-best set, rather than on the full search space which is usually exponentially large.

Previous algorithms for k-best parsing (Collins, 2000; Charniak and Johnson, 2005) are either sub-optimal or slow and rely significantly on pruning techniques to make them tractable. So I co-developed several fast and exact algorithms for k-best parsing in the general framework of directed monotonic hypergraphs (Huang and Chiang, 2005). This formulation extends and refines Klein and Manning's work (2001) by introducing *monotonic weight functions*, which is closely related to the optimal subproblem property in dynamic programming.

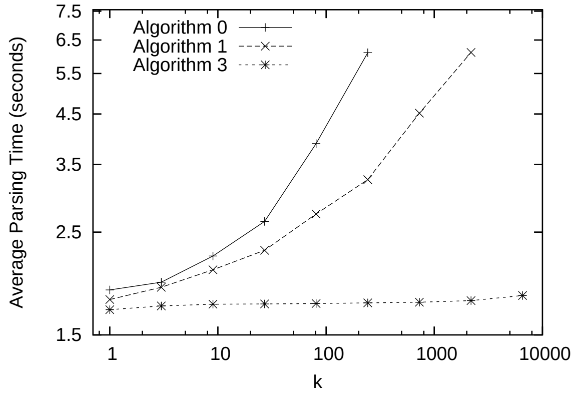


Figure 1: Average parsing speed on the Section 23 of Penn Treebank (Algorithms 0, 1, and 3, log-log).

We first generalize the classical 1-best Viterbi algorithm to hypergraphs, and then present four k-best algorithms, each improving its predessor by delaying more work until necessary. The final one, Algorithm 3, starts with a normal 1-best search for each vertex (or *item*, as in deductive frameworks), and then works backwards from the target vertex (final item) for its 2nd, 3rd, ..., kth best derivations, calling itself recursively only on demand, being the laziest of the four algorithms. When tested on top of two state-of-the-art systems, the Collins/Bikel parser (Bikel, 2004) and Chiang's CKY-based Hiero decoder (Chiang, 2005), this algorithm is shown to have very little overhead even for quite large k (say, 10^6) (See Fig. 1 for experiments on Bikel parser).

These algorithms have been re-implemented by other researchers in the field, including Eugene Charniak for his n-best parser, Ryan McDonald for his dependency parser (McDonald et al., 2005), Microsoft Research NLP group (Simon Corston-Oliver and Kevin Duh, p.c.) for a similar model, Jonathan Graehl for the ISI syntax-based MT decoder, David A. Smith for the Dyna language (Eisner et al., 2005), and Jonathan May for ISI's tree automata package Tiburon. All of these experiments confirmed the findings in our work.

2 Synchronous Binarization for MT

Machine Translation has made very good progress in recent times, especially, the so-called "phrase-based" statistical systems (Och and Ney, 2004). In order to take a substantial next-step it will be necessary to incorporate several aspects of syntax. Many researchers have explored syntax-based methods, for instance, Wu (1996) and Chiang (2005) both uses binary-branching synchronous context-free grammars (SCFGs). However, to be more expressive and flexible, it is often easier to start with a general SCFG or tree-transducer (Galley et al., 2004). In this case, *binarization* of the input grammar is required for the use of the CKY algorithm (in order to get cubic-time complexity), just as we convert a CFG into the Chomsky Normal Form (CNF) for monolingual parsing. For synchronous grammars, however, different binarization schemes may result in very different-looking chart items that greatly affect decoding efficiency. For example, consider the following SCFG rule:

(1) $\mathrm{S} \rightarrow \mathrm{NP}^{(1)}\ \mathrm{VP}^{(2)}\ \mathrm{PP}^{(3)},\ \mathrm{NP}^{(1)}\ \mathrm{PP}^{(3)}\ \mathrm{VP}^{(2)}$

We can binarize it either left-to-right or right-to-left:

$$\begin{array}{rl} \mathrm{S} \rightarrow & V_{\text{NP-PP}}\ \mathrm{VP} \\ V_{\text{NP-PP}} \rightarrow & \mathrm{NP}\ \mathrm{PP} \end{array} \quad \text{or} \quad \begin{array}{rl} \mathrm{S} \rightarrow & \mathrm{NP}\ V_{\text{PP-VP}} \\ V_{\text{PP-VP}} \rightarrow & \mathrm{PP}\ \mathrm{VP} \end{array}$$

The intermediate symbols (e.g. $V_{\text{PP-VP}}$) are called *virtual nonterminals*. We would certainly prefer the right-to-left binarization because the virtual nonterminal has consecutive span (see Fig. 2). The left-to-right binarization causes discontinuities on the target side, which results in an exponential time complexity when decoding with an integrated n-gram model.

We develop this intuition into a technique called *synchronous binarization* (Zhang et al., 2006) which binarizes a synchronous production or tree-tranduction rule on both source and target sides *simultaneously*. It essentially converts an SCFG into an equivalent ITG (the synchronous extension of CNF) if possible. We reduce this problem to the binarization of the permutation of nonterminal symbols between the source and target sides of a synchronous rule and devise a linear-time algorithm

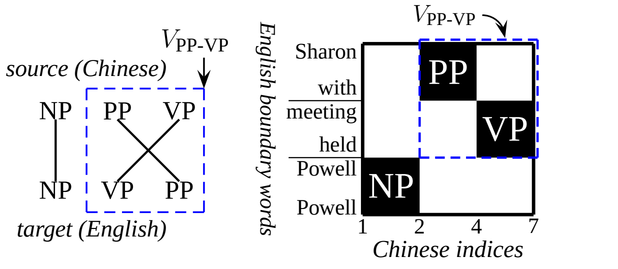


Figure 2: The alignment pattern (left) and alignment matrix (right) of the SCFG rule.

system	BLEU
monolingual binarization	36.25
synchronous binarization	*38.44*

Table 1: Synchronous vs. monolingual binarization in terms of translation quality (BLEU score).

for it. Experiments show that the resulting rule set significantly improves the speed and accuracy over monolingual binarization (see Table 1) in a state-of-the-art syntax-based machine translation system (Galley et al., 2004). We also propose another trick (hook) for further speeding up the decoding with integrated n-gram models (Huang et al., 2005).

3 Syntax-Directed Translation

Syntax-directed translation was originally proposed for compiling programming languages (Irons, 1961; Lewis and Stearns, 1968), where the source program is parsed into a syntax-tree that guides the generation of the object code. These translations have been formalized as a synchronous context-free grammar (SCFG) that generates two languages simultaneously (Aho and Ullman, 1972), and equivalently, as a top-down tree-to-string transducer (Gécseg and Steinby, 1984). We adapt this syntax-directed transduction process to statistical MT by applying stochastic operations at each node of the source-language parse-tree and searching for the best derivation (a sequence of translation steps) that converts the whole tree into some target-language string (Huang et al., 2006).

3.1 Extended Domain of Locality

From a modeling perspective, however, the structural divergence across languages results in non-isomorphic parse-trees that are not captured by

SCFGs. For example, the S(VO) structure in English is translated into a VSO order in Arabic, an instance of *complex re-ordering* (Fig. 4).

To alleviate this problem, grammars with richer expressive power have been proposed which can grab larger fragments of the tree. Following Galley et al. (2004), we use an extended tree-to-string transducer (**xRs**) with multi-level left-hand-side (LHS) trees.[1] Since the right-hand-side (RHS) string can be viewed as a flat one-level tree with the same nonterminal root from LHS (Fig. 4), this framework is closely related to STSGs in having extended domain of locality on the source-side except for remaining a CFG on the target-side. These rules can be learned from a parallel corpus using English parse-trees, Chinese strings, and word alignment (Galley et al., 2004).

3.2 A Running Example

Consider the following English sentence and its Chinese translation (note the reordering in the passive construction):

(2) the gunman was killed by the police .

qiangshou bei *jingfang jibi* 。
[gunman] [passive] [police] [killed] .

Figure 3 shows how the translator works. The English sentence (a) is first parsed into the tree in (b), which is then recursively converted into the Chinese string in (e) through five steps. First, at the root node, we apply the rule r_1 which preserves the top-level word-order and translates the English period into its Chinese counterpart:

(r_1) S (x_1:NP-C x_2:VP PUNC (.)) $\rightarrow x_1\ x_2$ 。

Then, the rule r_2 grabs the whole sub-tree for "the gunman" and translates it as a phrase:

(r_2) NP-C (DT (the) NN (gunman)) $\rightarrow$ *qiangshou*

Now we get a "partial Chinese, partial English" sentence "*qiangshou* VP 。" as shown in Fig. 3 (c). Our recursion goes on to translate the VP sub-tree. Here we use the rule r_3 for the passive construction:

[1]we will use LHS and source-side interchangeably (so are RHS and target-side). In accordance with our experiments, we also use English and Chinese as the source and target languages, opposite to the Foreign-to-English convention of Brown et al. (1993).

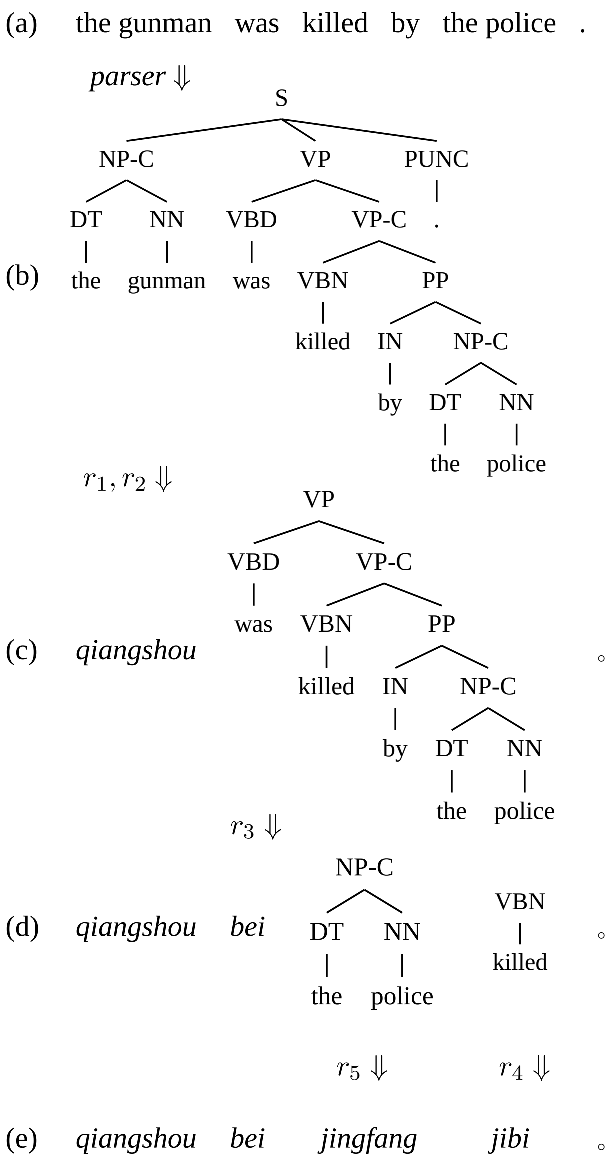


Figure 3: A synatx-directed translation process.

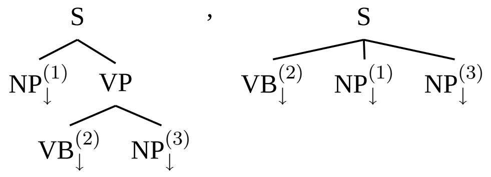
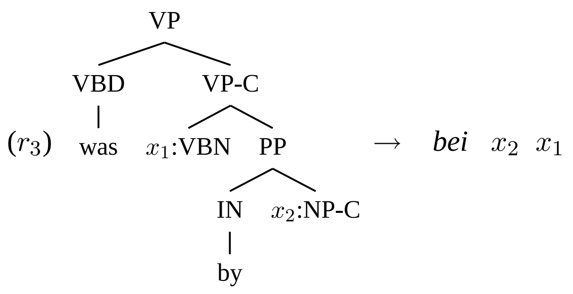


Figure 4: An example of complex re-ordering.



which captures the fact that the agent (NP-C, “the police”) and the verb (VBN, “killed”) are always inverted between English and Chinese in a passive voice. Finally, we apply rules r_4 and r_5 which perform phrasal translations for the two remaining subtrees in (d), respectively, and get the completed Chinese string in (e).

3.3 Translation Algorithm

Given a fixed parse-tree τ^*, the search for the best derivation (as a sequence of conversion steps) can be done by a simple top-down traversal (or depth-first search) from the root of the tree. With memoizationm, we get a dynamic programming algorithm that is guaranteed to run in $O(n)$ time where n is the length of the input string, since the size of the parse-tree is proportional to n. Similar algorithms have also been proposed for dependency-based translation (Lin, 2004; Ding and Palmer, 2005).

I am currently performing large-scale experiments on English-to-Chinese translation using the **xRs** rules. We are not doing the usual direction of Chinese-to-English partly due to the lack of a sufficiently good Chinese parser. Initial results show promising translation quality (in terms of BLEU scores) and fast translation speed.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*, volume I: Parsing of *Series in Automatic Computation*. Prentice Hall, Englewood Cliffs, New Jersey.

Daniel M. Bikel. 2004. Intricacies of Collins’ parsing model. *Computational Linguistics*, 30(4):479–511, December.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine-grained n-best parsing and discriminative reranking. In *Proceedings of the 43rd ACL*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of the 43rd ACL*.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of ICML*, pages 175–182.

Yuan Ding and Martha Palmer. 2005. Machine translation using probablisitic synchronous dependency insertion grammars. In *Proceedings of the 43rd ACL*.

Jason Eisner, Eric Goldlust, and Noah A. Smith. 2005. Compiling comp ling: Weighted dynamic programming and the dyna language. In *Proceedings of HLT-EMNLP*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What’s in a translation rule? In *HLT-NAACL*.

F. Gécseg and M. Steinby. 1984. *Tree Automata*. Akadémiai Kiadó, Budapest.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Liang Huang and David Chiang. 2005. Better k-best Parsing. In *Proceedings of 9th International Workshop of Parsing Technologies (IWPT)*.

Liang Huang, Hao Zhang, and Daniel Gildea. 2005. Machine translation as lexicalized parsing with hooks. In *Proceedings of 9th International Workshop of Parsing Technologies (IWPT)*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Syntax-directed translation with extended domain of locality. In submission.

E. T. Irons. 1961. A syntax-directed compiler for ALGOL 60. *Comm. ACM*, 4(1):51–55.

Dan Klein and Christopher D. Manning. 2001. Parsing and Hypergraphs. In *Proceedings of the Seventh International Workshop on Parsing Technologies (IWPT-2001), 17-19 October 2001, Beijing, China*.

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

Dekang Lin. 2004. A path-based transfer model for machine translation. In *Proceedings of the 20th COLING*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd ACL*.

F. J. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30:417–449.

Franz Och. 2003. Minimum error rate training for statistical machine translation. In *Proc. of ACL*.

Dekai Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *Proceedings of the 34th ACL*.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of HLT-NAACL*.