Learning Stochastic Bracketing Inversion Transduction Grammars with a Cubic Time Biparsing Algorithm

Markus SAERS **Joakim NIVRE**
Dept. of Linguistics and Philology
Uppsala University
Sweden
first.last@lingfil.uu.se

Dekai WU
Human Language Technology Center
Dept. of Computer Science and Engineering
HKUST
Hong Kong
dekai@cs.ust.hk

Abstract

We present a biparsing algorithm for Stochastic Bracketing Inversion Transduction Grammars that runs in $O(bn^3)$ time instead of $O(n^6)$. Transduction grammars learned via an EM estimation procedure based on this biparsing algorithm are evaluated directly on the translation task, by building a phrase-based statistical MT system on top of the alignments dictated by Viterbi parses under the induced bigrammars. Translation quality at different levels of pruning are compared, showing improvements over a conventional word aligner even at heavy pruning levels.

1 Introduction

As demonstrated by Saers & Wu (2009) there is something to be gained by applying structural models such as Inversion Transduction Grammars (ITG) to the problem of word alignment. One issue is that naïve methods for inducing ITGs from parallel data can be very time consuming. We introduce a parsing algorithm for inducing Stochastic Bracketing ITGs from parallel data in $O(bn^3)$ time instead of $O(n^6)$, where b is a pruning parameter (lower = tighter pruning). We try out different values for b, and evaluate the results on a translation tasks.

In section 2 we summarize the ITG framework; in section 3 we present our algorithm, whose time complexity is analyzed in section 4. In section 5 we describe how the algorithm is evaluated, and in section 6, the empirical results are given.

2 Inversion Transduction Grammars

Inversion transductions are a theoretically interesting and empirically useful equivalence class of transductions, with expressiveness and computational complexity characteristics lying intermediate between finite-state transductions and syntax-directed transductions. An Inversion Transduction Grammar (ITG) can be used to synchronously generate sentence pairs, synchronously parse sentence pairs, or transduce from a sentence in one language to a sentence in another.[1]

The equivalence class of inversion transductions can be described by restricting Syntax-Directed Transduction Grammars (SDTG)[2] in various equivalent ways to the special cases of (a) binary SDTGs, (b) ternary SDTGs, or (c) SDTGs whose transduction rules are restricted to straight and inverted permutations only.

Thus on one hand, any binary or ternary SDTG is an ITG. Conversely, any ITG can be stated in binary two-normal form (Wu, 1997). Only three kinds of rules are present in the normal form:

$$A \to [BC]$$
$$A \to \langle BC \rangle$$
$$A \to e/f$$

On the other hand, under characterization (c), what distinguishes ITGs is that the permutation of constituents is restricted in such a way that all children of a node must be read either left-to-right, or right-to-left. The movement only applies to one of the languages, the other is fixed. Formally, an ITG is a tuple $\langle N, V, \Delta, S \rangle$, where N is a set of nonterminal symbols, Δ is a set of rewrite rules, $S \in N$ is the start symbol and $V \subseteq V_E \times V_F$ is a set of biterminal symbols, where V_E is the vocabulary of E and V_F is the vocabulary of F. We will write a biterminal as e/f, where $e \in V_E$ and $f \in V_F$. A sentence pair will be written as $\mathbf{e}/\mathbf{f}$, and a bispan as $e_{s..t}/f_{u..v}$.

Each rule $\delta \in \Delta$ is a tuple $\langle X, \gamma, \theta \rangle$ where $X \in N$ is the right hand side of the rule, $\gamma \in$

[1] All transduction grammars (a.k.a. synchronous grammars, or simply bigrammars) can be interpreted as models for generation, recognition, or transduction.

[2] SDTGs (Lewis & Stearns (1968); Aho & Ullman (1969), (1972)) are also recently called synchronous CFGs.

Proceedings of the 11th International Conference on Parsing Technologies (IWPT), pages 29–32,
Paris, October 2009.

$\{N \cup V\}^*$ is a series of nonterminal and biterminal symbols representing the production of the rule and $\theta \in \{\emptyset, [], \langle\rangle\}$ denotes the orientation (axiomatic, straight or inverted) of the rule. Straight rules are read left-to-right in both languages, while inverted rules are read left-to-right in E and right-to-left in F. The direction of the axiomatic rules is undefined, as they must be completely made up of terminals. For notational convenience, the orientation of the rule is written as surrounding the production, like so: $X \rightarrow \gamma$, $X \rightarrow [\gamma]$ and $X \rightarrow \langle\gamma\rangle$. The vocabularies of the languages may both include the empty token ϵ, allowing for deletions and insertions. The empty biterminal, ϵ/ϵ is not allowed.

2.1 Stochastic ITGs

In a Stochastic ITG (SITG), each rule is also associated with a probability, such that

$$\sum_{\gamma} Pr(X \rightarrow \gamma) = 1$$

for all $X \in N$. The probability of a derivation $S \stackrel{*}{\Rightarrow} \mathbf{e}/\mathbf{f}$ is defined as the production of the probabilities of all rules used. As shown by Wu (1995), it is possible to fit the parameters of a SITG to a parallel corpus via EM (expectation-maximization) estimation.

2.2 Bracketing ITGs

An ITG where there is only one nonterminal (other than the start symbol) is called a bracketing ITG (BITG). Since the one nonterminal is devoid of information, it can only be used to group its children together, imposing a bracketing on the sentence pairs.

3 Parsing SBITGs

In this section we present a biparsing algorithm for Stochastic Bracketing Inversion Transduction Grammars (SBITGs) in normal form which incorporates a pruning parameter b. The algorithm is basically an agenda-based bottom-up chart parser, where the pruning parameter controls the number of active items of a given length.

To parse a sentence pair $\mathbf{e}/\mathbf{f}$, the parser needs a chart $\mathcal{C}$ and a series of $T + V$ agendas $A_1, A_2, \ldots, A_{T+V}$, where $T = |\mathbf{e}|$ and $V = |\mathbf{f}|$. An item is defined as a nonterminal symbol (we use X to denote the anonymous nonterminal symbol of the bracketing ITG) and one span in each language, written as X_{stuv} where $0 \leq s \leq t \leq T$ corresponds to the span $e_{s..t}$ and $0 \leq u \leq v \leq V$ corresponds to the span $f_{u..v}$. The length of an item is defined as $|X_{stuv}| = (t-s)+(v-u)$. Since items are grouped by their length, highly skewed links (eg. 6:1) will be competing with very even links (eg. 4:3). Skewed links are generally bad (and should be pruned), or have a high probability (which means they are likely to survive pruning). An item may be active or passive, the active items are present in the agendas and the chart, whereas the passive items are only present in the chart.

The parser starts by asserting items from all lexical rules ($X \rightarrow e/f$), and placing them on their respective agendas. After the initial seeding, the agendas are processed in order. When an agenda is processed, it is first pruned, so that only the b best items are kept active. After pruning, the remaining active items are allowed to be extended. When extended, the item combines with an adjacent item in the chart to form a larger item. The newly created item is considered active, and added to both the chart and the appropriate agenda. Once an item has been processed it goes from being active to being passive. The process is halted when the goal item S_{0T0V} is reached, or when no active items remain. To build the forest corresponding to the parse process, back-pointers are used.

3.1 Initialization

In the initial step, the set of lexical items L is built. All lexical items $i \in L$ are then activated by placing them on their corresponding agenda $A_{|i|}$.

$$L = \left\{ X_{stuv} \;\middle|\; \begin{array}{l} 0 \leq s \leq t \leq T, \\ 0 \leq u \leq v \leq V, \\ X \rightarrow e_{s..t}/f_{u..v} \in \Delta \end{array} \right\}$$

By limiting the length of phrasal terminals to some threshold μ, the variables t and v can be limited to $s+\mu$ and $u+\mu$ respectively, limiting the complexity of the initialization step from $O(n^4)$ to $O(n^2)$.

3.2 Recursion

In the recursive step we build a set of extensions $E(i)$ for all active items i. All items in $E(i)$ are then activated by placing them on their corresponding agenda ($i \in A_{|i|}$).

$$\begin{array}{l} E(X_{stuv}) = \\ \{X_{StUv} | 0 \leq S \leq s, 0 \leq U \leq u, X_{SsUu} \in \mathcal{C}\} \quad \cup \\ \{X_{sSuU} | t \leq S \leq T, v \leq U \leq V, X_{tSvU} \in \mathcal{C}\} \quad \cup \\ \{X_{sSUv} | t \leq S \leq T, 0 \leq U \leq u, X_{tSUu} \in \mathcal{C}\} \quad \cup \\ \{X_{StuU} | 0 \leq S \leq s, v \leq U \leq V, X_{SsvU} \in \mathcal{C}\} \end{array}$$

Since we are processing the agendas in order, any item in the chart will be as long as or shorter than the item being extended. This fact can be exploited to limit the number of possible siblings explored, but has no impact on time complexity.

3.3 Viterbi parsing

When doing Viterbi parsing, all derivations but the most probable are discarded. This gives an unambiguous parse, which dictates exactly one alignment between $\mathbf{e}$ and $\mathbf{f}$. The alignment of the Viterbi parse can be used to substitute that of other word aligners (Saers and Wu, 2009) such as GIZA++ (Och and Ney, 2003).

4 Analysis

Looking at the algorithm, it is clear that there will be a total of $T + V = O(n)$ agendas, each containing items of a certain length. The items in an agenda can start anywhere in the alignment space: $O(n^2)$ possible starting points, but once the end point in one language is set, the end point in the other follows from that, adding a factor $O(n)$. This means that each agenda contains $O(n^3)$ active items. Each active item has to go through all possible siblings in the recursive step. Since the start point of the sibling is determined by the item itself (it has to be adjacent), only the $O(n^2)$ possible end points have to be explored. This means that each active item takes $O(n^2)$ time to process.

The total time is thus $O(n^6)$: $O(n)$ agendas, containing $O(n^3)$ active items, requiring $O(n^2)$ time to process. This is also the time complexity reported for ITGs in previous work (Wu, 1995; Wu, 1997).

The pruning works by limiting the number of active items in an agenda to a constant b, meaning that there are $O(n)$ agendas, containing $O(b)$ active items, requiring $O(n^2)$ time to process. This gives a total time complexity of $O(bn^3)$.

5 Evaluation

We evaluate the parser on a translation task (WMT'08 shared task[3]). In order to evaluate on a translation task, a translation system has to be built. We use the alignments from the Viterbi parses of the training corpus to substitute the alignments of GIZA++. This is the same approach as taken in Saers & Wu (2009). We will evaluate the resulting translations with two automatic metrics: BLEU (Papineni et al., 2002) and NIST (Doddington, 2002).

[3] `http://www.statmt.org/wmt08/`

6 Empirical results

In this section we describe the experimental setup as well as the outcomes.

6.1 Setup

We use the Moses Toolkit (Koehn et al., 2007) to train our phrase-based SMT models. The toolkit also includes scripts for applying GIZA++ (Och and Ney, 2003) as a word aligner. We have trained several systems, one using GIZA++ (our baseline system), one with no pruning at all, and 6 different values of b (1, 10, 25, 50, 75 and 100). We used the `grow-diag-final-and` method to extract phrases from the word alignment, and MERT (Och, 2003) to optimize the resulting model. We trained a 5-gram SRI language model (Stolcke, 2002) using the corpus supplied for this purpose by the shared task organizers. All of the above is consistent with the guidelines for building a baseline system for the WMT'08 shared task.

The translation tasks we applied the above procedure to are all taken from the Europarl corpus (Koehn, 2005). We selected the tasks German-English, French-English and Spanish-English. Furthermore, we restricted the training sentence pairs so that none of the sentences exceeded length 10. This was necessary to be able to carry out exhaustive search. The total amount of training data was roughly 100,000 sentence pairs in each language pair, which is a relatively small corpus, but by no means a toy example.

6.2 Grammar induction

It is possible to set the parameters of a SBITG by applying EM to an initial guess (Wu, 1995). As our initial guess, we used word co-occurrence counts, assuming that there was one empty token in each sentence. This gave an estimate of the lexical rules. The probability mass was divided so that the lexical rules could share half of it, while the other half was shared equally by the two structural rules ($X \to [XX]$ and $X \to \langle XX \rangle$).

Several training runs were made with different pruning parameters. The EM process was halted when a relative improvement in log-likelihood of 10^{-3} was no longer achieved over the previous iteration.

Metric	Baseline (GIZA++)	Different values of b for SBITGs						
		∞	100	75	50	25	10	1
Spanish-English								
BLEU	0.2597	0.2663	**0.2671**	0.2661	0.2653	0.2655	0.2608	0.1234
NIST	6.6352	6.7407	**6.7445**	6.7329	6.7101	6.7312	6.6439	3.9705
time		03:20:00	02:40:00	02:00:00	01:20:00	00:38:00	00:17:00	00:03:10
German-English								
BLEU	0.2059	**0.2113**	0.2094	0.2091	0.2090	0.2091	0.2050	0.0926
NIST	5.8668	**5.9380**	5.9086	5.8955	5.8947	5.9292	5.8743	3.4297
time		03:40:00	02:45:00	02:10:00	01:25:00	00:41:00	00:17:00	00:03:20
French-English								
BLEU	0.2603	0.2663	0.2655	0.2668	**0.2669**	0.2654	0.2632	0.1268
NIST	6.6907	**6.8151**	6.8068	6.8068	6.8065	6.7013	6.7136	4.0849
time		03:10:00	02:45:00	02:10:00	01:25:00	00:42:00	00:17:00	00:03:25

Table 1: Results. Time measures are approximate time per iteration.

Once the EM process terminated, Viterbi parses were calculated for the training corpus, and the alignments from them outputted in the same format produced by GIZA++.

6.3 Results

The results are presented in Table 1. GIZA++ generally terminates within minutes (6–7) on the training corpora used, making it faster than any of the SBITGs (they generally required 4–6 iterations to terminate, making even the fastest ones slower than GIZA++). To put the times in perspective, about 6 iterations were needed to get the ITGs to converge, making the longest training time about 16–17 hours. The time it takes to extract the phrases and tune the model using MERT is about 14 hours for these data sets.

Looking at translation quality, we see a sharp initial rise as b grows to 10. At this point the SBITG system is on par with GIZA++. It continues to rise up to $b = 25$, but after that is more or less levels out. From this we conclude that the positive results reported in Saers & Wu (2009) hold under harsh pruning.

7 Conclusions

We have presented a SBITG biparsing algorithm that uses a novel form of pruning to cut the complexity of EM-estimation from $O(n^6)$ to $O(bn^3)$. Translation quality using the resulting learned SBITG models is improved over using conventional word alignments, even under harsh levels of pruning.

Acknowledgments

The authors are grateful for the comments made by the two anonymous reviewers. This work was funded by the Swedish National Graduate School of Language Technology, the Defense Advanced Research Projects Agency (DARPA) under GALE Contract No. HR0011-06-C-0023, and the Hong Kong Research Grants Council (RGC) under research grants GRF621008, DAG03/04.EG09, RGC6256/00E, and RGC6083/99E. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency.

References

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

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling (Volumes 1 and 2)*. Prentice-Halll, Englewood Cliffs, NJ.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Human Language Technology conference (HLT-2002)*, San Diego, CA.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL-2007 Demo and Poster Sessions*, pages 177–180, Prague, Jun.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Machine Translation Summit X*, Phuket, Thailand, September.

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

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–52.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, Jul.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translations. In *40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pages 311–318, Philadelphia, Jul.

Markus Saers and Dekai Wu. 2009. Improving phrase-based translation via word alignments from Stochastic Inversion Transduction Grammars. In *Proceedings of SSST-3, Third Workshop on Syntax and Structure in Statistical Translation (at NAACL HLT 2009)*, pages 28–36, Boulder, CO, Jun.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *International Conference on Spoken Language Processing*, Denver, CO, Sep.

Dekai Wu. 1995. Trainable coarse bilingual grammars for parallel text bracketing. In *Third Annual Workshop on Very Large Corpora (WVLC-3)*, pages 69–81, Cambridge, MA, Jun.

Dekai Wu. 1997. Stochastic Inversion Transduction Grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404, Sep.