Ternary Segmentation for Improving Search in Top-down Induction of Segmental ITGs

Markus Saers Dekai Wu
HKUST
Human Language Technology Center
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
{masaers|dekai}@cs.ust.hk

Abstract

We show that there are situations where iteratively segmenting sentence pairs top-down will fail to reach valid segments and propose a method for alleviating the problem. Due to the enormity of the search space, error analysis has indicated that it is often impossible to get to a desired embedded segment purely through binary segmentation that divides existing segmental rules in half – the strategy typically employed by existing search strategies – as it requires two steps. We propose a new method to hypothesize ternary segmentations in a single step, making the embedded segments immediately discoverable.

1 Introduction

One of the most important improvements to statistical machine translation to date was the move from token-based model to segmental models (also called phrasal). This move accomplishes two things: it allows a flat surface-based model to memorize some relationships between word realizations, but more importantly, it allows the model to capture multi-word concepts or chunks. These chunks are necessary in order to translate fixed expressions, or other multi-word units that do not have a *compositional meaning*. If a sequence in one language can be broken down into smaller pieces which are then translated individually and reassembled in another language, the meaning of the sequence is compositional; if not, the only way to translate it accurately is to treat it as a single unit – a chunk. Existing surface-based models (Och *et al.*, 1999) have high recall in capturing the chunks, but tend to over-generate, which leads to big models and low precision. Surface-based models have no concept of hierarchical composition, instead they make the assumption that a sentence consists of a sequence of segments that can be individually translated and reordered to form the translation. This is counter-intuitive, as the *who*-did-*what*-to-*whom*s of a sentence tends to be translated and reordered as units, rather than have their components mixed together. Transduction grammars (Aho and Ullman, 1972; Wu, 1997), also called hierarchical translation models (Chiang, 2007) or synchronous grammars, address this through a mechanism similar to context-free grammars. Inducing a segmental transduction grammar is hard, so the standard practice is to use a similar method as the surface-based models use to learn the chunks, which is problematic, since that method mostly relies on memorizing the relationships that the mechanics of a compositional model is designed to generalize. A compositional translation model would be able to translate lexical chunks, as well as generalize different kinds of compositions; a segmental transduction grammar captures this by having segmental lexical rules and different nonterminal symbols for different categories of compositions. In this paper, we focus on inducing the former: segmental lexical rules in inversion transduction grammars (ITGs).

One natural way would be to start with a token-based grammar and chunk adjacent tokens to form segments. The main problem with chunking is that the data becomes more and more likely as the segments get larger, with the degenerate end point of all sentence pairs being memorized lexical items. Zhang *et al.* (2008) combat this tendency by introducing a sparsity prior over the rule probabilities, and variational Bayes to maximize the posterior probability of the data subject to this symmetric Dirichlet prior. To hypothesize possible chunks, they examine the Viterbi biparse of the existing model. Saers *et al.* (2012) use the entire parse forest to generate the hypotheses. They also bootstrap the ITG from linear and finite-state transduction grammars (LTGs, Saers (2011), and FSTGs),

Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, pages 86–93,
October 25, 2014, Doha, Qatar.

rather than initialize the lexical probabilities from IBM models.

Another way to arrive at a segmental ITG is to start with the degenerate chunking case: each sentence pair as a lexical item, and segment the existing lexical rules into shorter rules. Since the start point is the degenerate case when optimizing for data likelihood, this approach requires a different objective function to optimize against. Saers *et al.* (2013c) proposes to use description length of the model and the data given the model, which is subsequently expressed in a Bayesian form with the addition of a prior over the rule probabilities (Saers and Wu, 2013). The way they generate hypotheses is restricted to segmenting an existing lexical item into two parts, which is problematic, because embedded lexical items are potentially overlooked.

There is also the option of implicitly defining all possible grammars, and sample from that distribution. Blunsom *et al.* (2009) do exactly that; they induce with collapsed Gibbs sampling which keeps one derivation for each training sentence that is altered and then resampled. The operations to change the derivations are **split**, **join**, **delete** and **insert**. The split-operator corresponds to binary segmentation, the join-operator corresponds to chunking; the delete-operator removes an internal node, resulting in its parent having three children, and the insert-operator allows a parent with three children to be normalized to have only two. The existence of ternary nodes in the derivation means that the learned grammar contains ternary rules. Note that it still takes three operations: *two split-operations and one delete-operation* for their model to do what we propose to do in a single ternary segmentation. Also, although we allow for single-step ternary segmentations, our grammar does not contain ternary rules; instead the results of a ternary segmentation is immediately normalized to the 2-normal form. Although their model can *theoretically* sample from the entire model space, the split-operation alone is enough to do so; the other operations were added to get the model to do so in *practice*. Similarly, we propose ternary segmentation to be able to reach areas of the model space that we failed to reach with binary segmentation.

To illustrate the problem with embedded lexical items, we will introduce a small example corpus. Although Swedish and English are relatively similar, with the structure of basic sentences being identical, they already illustrate the common problem of rare embedded correspondences. Imagine a really simple corpus of three sentence pairs with identical structure:

he has a red book / han har en röd bok

she has a biology book / hon har en biologibok

it has begun / det har börjat

The main difference is that Swedish concatenates rather than juxtaposes compounds such as *biologibok* instead of *biology book*. A bilingual person looking at this corpus would produce bilingual parse trees like those in Figure 1. Inducing this relatively simple segmental ITG from the data is, however, quite a challenge.

The example above illustrates a problem with the chunking approach, as one of the most common chunks is *has a*/*har en*, whereas the linguistically motivated chunk *biology book*/*biologibok* occurs only once. There is very little in this data that would lead the chunking approach towards the desired ITG. It also illustrates a problem with the binary segmentation approach, as all the bilingual prefixes and suffixes, the **biaffixes**, are unique; there is no way of discovering that all the above sentences have the exact same verb.

In this paper, we propose a method to allow bilingual infixes to be hypothesized and used to drive the minimization of description length, which would be able to induce the desired ITG from the above corpus.

The paper is structured so that we start by giving a definition of the grammar formalism we use: ITGs (Section 2). We then describe the notion of description length that we use (Section 3), and how ternary segmentation differs from and complements binary segmentation (Section 4). We then present our induction algorithm (Section 5) and give an example of a run through (Section 6). Finally we offer some concluding remarks (Section 7).

2 Inversion transduction grammars

Inversion transduction grammars, or ITGs (Wu, 1997), are an expressive yet efficient way to model translation. Much like context-free grammars (CFGs), they allow for sentences to be explained through composition of smaller units into larger units, but where CFGs are restricted to generate monolingual sentences, ITGs generate sets of sentence pairs – **transductions** – rather than languages. Naturally, the components of differ-

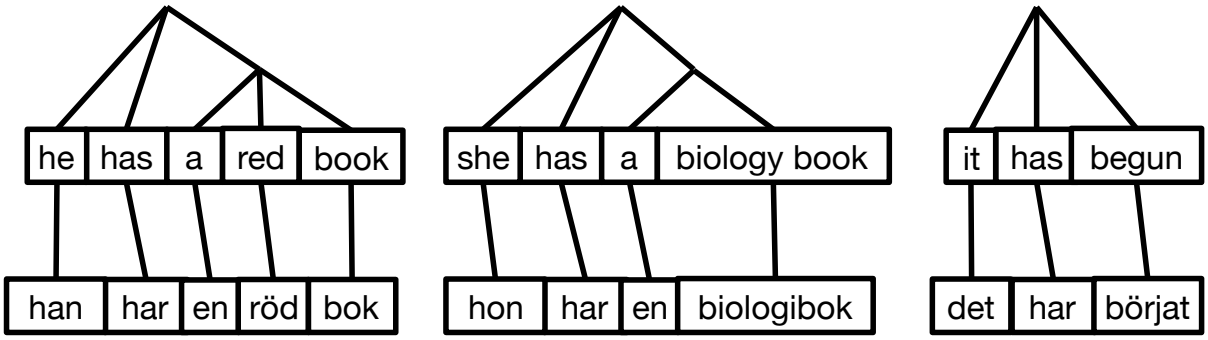

Figure 1: Possible inversion transduction trees over the example sentence pairs.

ent languages may have to be ordered differently, which means that transduction grammars need to handle these differences in order. Rather than allowing arbitrary reordering and pay the price of exponential time complexity, ITGs allow only monotonically straight or inverted order of the productions, which cuts the time complexity down to a manageable polynomial.

Formally, an ITG is a tuple $\langle N, \Sigma, \Delta, R, S \rangle$, where N is a finite nonempty set of nonterminal symbols, Σ is a finite set of terminal symbols in L_0, Δ is a finite set of terminal symbols in L_1, R is a finite nonempty set of inversion transduction rules and $S \in N$ is a designated start symbol. An inversion transduction rule is restricted to take one of the following forms:

$$S \to [A]\,,\ A \to \left[\psi^+\right]\,,\ A \to \langle \psi^+ \rangle$$

where $S \in N$ is the start symbol, $A \in N$ is a nonterminal symbol, and ψ^+ is a nonempty sequence of nonterminals and biterminals. A **biterminal** is a pair of symbol strings: $\Sigma^* \times \Delta^*$, where at least one of the strings have to be nonempty. The square and angled brackets signal straight and inverted order respectively. With straight order, both the L_0 and the L_1 productions are generated left-to-right, but with inverted order, the L_1 production is generated right-to-left. The brackets are frequently left out when there is only one element on the right-hand side, which means that $S \to [A]$ is shortened to $S \to A$.

Like CFGs, ITGs also have a 2-normal form, analogous to the Chomsky normal form for CFGs, where the rules are further restricted to only the following four forms:

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

where $S \in N$ is the start symbol, $A, B, C \in N$ are nonterminal symbols and e/f is a biterminal string.

A bracketing ITG, or BITG, has only one nonterminal symbol (other than the dedicated start symbol), which means that the nonterminals carry no information at all other than the fact that their yields are discrete unit. Rather than make a proper analysis of the sentence pair they only produce a bracketing, hence the name.

A transduction grammar such as ITG can be used in three modes: **generation**, **transduction** and **biparsing**. Generation derives a **bisentence**, a sentence pair, from the start symbol. Transduction derives a sentence in one language from a sentence in the other language and the start symbol. Biparsing verifies that a given bisentence can be derived from the start symbol. Biparsing is an integral part of any learning that requires expected counts such as expectation maximization, and transduction is the actual translation process.

3 Description length

We follow the definition of description length from Saers *et al.* (2013b,c,d,a); Saers and Wu (2013), that is: the size of the model is determined by counting the number of symbols needed to encode the rules, and the size of the data given the model is determined by biparsing the data with the model. Formally, given a grammar Φ its description length $\mathrm{DL}(\Phi)$ is the sum of the length of the symbols needed to serialize the rule set. For convenience later on, the symbols are assumed to be uniformly distributed with a length of $-\lg \frac{1}{N}$ bits each (where N is the number of different symbols). The description length of the data D given the model is defined as $\mathrm{DL}(D|\Phi) = -\lg P(D|\Phi)$.

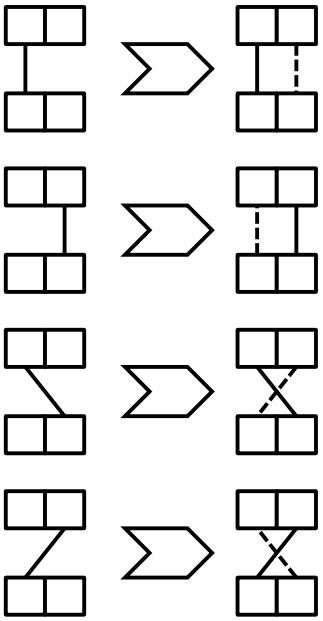

Figure 2: The four different kinds of binary segmentation hypotheses.

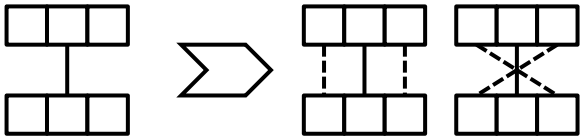

Figure 3: The two different hypotheses that can be made from an infix-to-infix link.

4 Segmenting lexical items

With a background in computer science it is tempting to draw the conclusion that any segmentation can be made as a sequence of binary segmentations. This is true, but only relevant if the entire search space can be *exhaustively* explored. When inducing transduction grammars, the search space is prohibitively large; in fact, we are typically afforded only an estimate of a single step forward in the search process. In such circumstances, the kinds of steps you can take start to matter greatly, and adding ternary segmentation to the typically used binary segmentation adds expressive power.

Figure 2 contains a schematic illustration of binary segmentation: To the left is a lexical item where a good biaffix (an L_0 prefix or suffix associated with an L_1 prefix or suffix) has been found, as illustrated with the solid connectors. To the right is the segmentation that can be inferred. For binary segmentation, there is no uncertainty in this step.

When adding ternary segmentation, there are five more situations: one situation where an infix is linked to an infix, and four situations where an infix is linked to an affix. Figure 3 shows the infix-to-infix situation, where there is one additional piece of information to be decided: are the surroundings linked straight or inverted? Figure 4 shows the situations where one infix is linked to an affix. In these situations, there are two more pieces of information that needs to be inferred: (a) where the sibling of the affix needs to be segmented, and (b) how the two pieces of the sibling of the affix link to the siblings of the infix. The infix-to-affix situations require a second monolingual segmentations decision to be made. As this is beyond the scope of this paper, we will limit ourselves to the infix-to-infix situation.

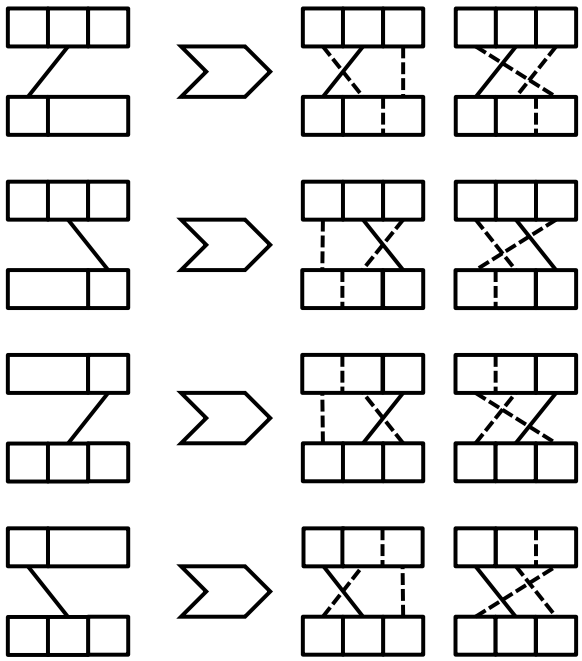

Figure 4: The eight different hypotheses that can be made from the four different infix-to-affix links.

5 Finding segmentation hypotheses

Previous work on binary hypothesis generation makes assumptions that do not hold with ternary segmentation; this section explains why that is and how we get around it. The basic problem with binary segmentation is that any bisegment hypothesized to be good on its own has to be anchored to either the beginning or the end of an existing bisegment. An infix, by definition, does not.

While recording all affixes is possible, even for non-toy corpora (Saers and Wu, 2013; Saers *et al.*, 2013b,c), recording all bilingual infixes is not, so collecting them all is not an option (while there are

Algorithm 1 Pseudo code for segmenting an ITG.

Φ ▷ The ITG being induced.
Ψ ▷ The token-based ITG used to evaluate lexical rules.
h_{max} ▷ The maximum number of hypotheses to keep from a single lexical rule.
repeat
  $\delta \leftarrow 0$
  H' ▷ Initial hypotheses
  for all lexical rules $A \rightarrow e/f$ **do**
    $p \leftarrow parse(\Psi, e/f)$
    c ▷ Fractional counts of bispans
    for all bispans $s, t, u, v \in e/f$ **do**
      $c(s,t,u,v) \leftarrow 0$
    $H'' \leftarrow []$
    for all items $B_{s,t,u,v} \in p$ **do**
      $c(s,t,u,v) \leftarrow c(s,t,u,v) + \alpha(B_{s,t,u,v})\beta(B_{s,t,u,v})/\alpha(S_{0,T,0,V})$
      $H'' \leftarrow [H'', \langle s,t,u,v,c(s,t,u,v)\rangle]$
    sort H'' on $c(s,t,u,v)$
    for all $\langle s,t,u,v,c(s,t,u,v)\rangle \in H''[0..h_{max}]$ **do**
      $H'(e_{s..t}/f_{u..v}) \leftarrow [H'(e_{s..t}/f_{u..v}), \langle s,t,u,v,A \rightarrow e/f\rangle]$
  H ▷ Evaluated hypotheses
  for all bisegments $e_{s..t}/f_{u..v} \in keys(H')$ **do**
    $\Phi' \leftarrow \Phi$
    $R \leftarrow []$
    for all bispan-rule pairs $\langle s,t,u,v,A \rightarrow e/f\rangle \in H'(e_{s..t}/f_{u..v})$ **do**
      $\Phi' \leftarrow make\_grammar\_change(\Phi', e/f, s, t, u, v)$
      $R \leftarrow [R, A \rightarrow e/f]$
    $\delta' \leftarrow DL(\Phi') - DL(\Phi) + DL(D|\Phi') - DL(D|\Phi)$
    if $\delta' < 0$ **then**
      $H \leftarrow [H, \langle e_{s..t}/f_{u..v}, R, \delta'\rangle]$
  sort H on δ'
  for all $\langle e_{s..t}/f_{u..v}, R, \delta'\rangle \in H$ **do**
    $\Phi' \leftarrow \Phi$
    for all rules $A \rightarrow e/f \in R \cap R_{\Psi'}$ **do**
      $\Phi' \leftarrow make\_grammar\_change(\Phi', e/f, s, t, u, v)$
    $\delta' \leftarrow DL(\Phi') - DL(\Phi) + DL(D|\Phi') - DL(D|\Phi)$
    if $\delta' < 0$ **then**
      $\Phi \leftarrow \Phi'$
      $\delta \leftarrow \delta + \delta'$
until $\delta \leq 0$
return Φ

only $O\left(n^2\right)$ possible biaffixes for a parallel sentence of average length n, there are $O\left(n^4\right)$ possible bilingual infixes). A way to prioritize, within the scope of a single bisegment, which infixes and affixes to consider as hypotheses is crucial. In this paper we use an approach similar to Saers *et al.* (2013d), in which we use a token-based ITG to evaluate the lexical rules in the ITG that is being induced. Using a transduction grammar has the advantage of calculating fractional counts for hypotheses, which allows both long and short hypotheses to compete on a level playing field.

In Algorithm 1, we start by parsing all the lexical rules in the grammar Φ being learned using a token-based ITG Ψ. For each rule, we only keep the best h_{max} bispans. In the second part, all collected bispans are evaluated as if they were the only hypothesis being considered for changing Φ. Any hypothesis with a positive effect is kept for further processing. These hypotheses are sorted

and applied. Since the grammar may have changed since the effect of the hypothesis was estimated, we have to check that the hypothesis would have a positive effect on the *updated* grammar before committing to it. All this is repeated as long as there are improvements that can be made.

The *make_grammar_change* method deletes the old rule, and distributes its probability mass to the rules replacing it. For ternary segmentation, this will be three lexical rules, and two structural rules (which happens to be identical in a bracketing grammar, giving that one rule two shares of the probability mass being distributed). For binary segmentation it is two lexical rules and one structural rule.

Rather than calculating $\mathrm{DL}\,(D|\Phi) - \mathrm{DL}\,(D|\Phi)$ explicitly by biparsing the entire corpus, we estimate the change. For binary rules, we use the same estimate as Saers and Wu (2013): multiplying in the new rule probabilities and dividing out the old. For ternary rules, we make the assumption that the three new lexical rules are combined using structural rules the way they would during parsing, which means two binary structural rules being applied. The infix-to-infix situation must be generated *either* by two straight combinations *or* by two inverted combinations, so for a bracketing grammar it is always two applications of a single structural rule. We thus multiply in the three new lexical rules and the structural rule twice, and divide out the old rule. In essence, both these methods are recreating the situations in which the parser would have used the old rule, but now uses the new rules.

Having exhausted all the hypotheses, we also run expectation maximization to stabilize the parameters. This step is not shown in the pseudo code.

Examining the pseudocode closer reveals that the outer loop will continue as long as the grammar changes; since the only way the grammar changes is by making lexical rules shorter, this loop is guaranteed to terminate. Inside the outer loop there are three inner loops: one over the rule set, one over the set of initial hypotheses H' and one over the set of evaluated hypotheses H. The sets of hypotheses are related such that $|H| \leq |H'|$, which means that the size of the initial set of hypotheses will dominate the time complexity. The size of this initial set of hypotheses is itself limited so that it cannot contain more than $h_{\max}$ hypotheses from any one rule. The dominating factor is thus the size of the rule set, which we will further analyze.

The first thing we do is to parse the right-hand side of the rule, which requires $O\left(n^3\right)$ with the Saers *et al.* (2009) algorithm, where n is the average length of the lexical items. We then initialize the counts, which does not actually require a specific step in implementation. We then iterate over all bispans in the parse, which has the same upper bound as the parsing process, since the approximate parsing algorithm avoids exploring the entire search space. We then sort the set of hypotheses derived from the current rule only, which is asymptotically bound by $O\left(n^3 \lg n\right)$, since there is exactly one hypothesis per parse item. Finally, there is a selection being made from the set of hypotheses derived from the current rule. In practice, the parsing is more complicated than the sorting, making the time complexity of the whole inner loop be dominated by the time it takes to parse the rules.

6 Example

In this section we will trace through how the example from the introduction fails to go through binary segmentation, but succeeds when infix-to-infix segmentations are an option.

The initial grammar consists of all the sentence pairs as segmental lexical rules:

$$
\begin{array}{rl}
S \to A & 1 \\
A \to \frac{\text{he has a red book}}{\text{han har en röd bok}} & 0.\overline{3} \\
A \to \frac{\text{she has a biology book}}{\text{hon har en biologibok}} & 0.\overline{3} \\
A \to \frac{\text{it has begun}}{\text{det har börjat}} & 0.\overline{3}
\end{array}
$$

As noted before, there are no shared biaffixes among the three lexical rules, so binary segmentation cannot break this grammar down further. There are, however, three shared bisegments representing three different segmentation hypotheses: *has a*/*har en*, *has*/*har* and *a*/*en*. In this example it does not matter which hypothesis you choose, so we will go with the first one, since that is the one our implementation chose. Breaking out all occurrences of *has a*/*har en* gives the following gram-

mar:

$$\begin{array}{cc} S \rightarrow A & 1 \\ A \rightarrow [AA] & 0.\overline{36} \\ A \rightarrow \text{it has begun/det har börjat} & 0.\overline{09} \\ A \rightarrow \text{has a/har en} & 0.\overline{18} \\ A \rightarrow \text{he/han} & 0.\overline{09} \\ A \rightarrow \text{red book/röd bok} & 0.\overline{09} \\ A \rightarrow \text{she/hon} & 0.\overline{09} \\ A \rightarrow \text{biology book/biologibok} & 0.\overline{09} \end{array}$$

At this point there are two bisegments that occur in more than one rule: *has*/*har* and *a*/*en*. Again, it does not matter for the final outcome which of the hypotheses we choose, so we will chose the first one, again because that is the one our implementation chose. Breaking out all occurrences of *has*/*har* gives the following grammar:

$$\begin{array}{cc} S \rightarrow A & 1 \\ A \rightarrow [AA] & 0.421 \\ A \rightarrow \text{he/han} & 0.053 \\ A \rightarrow \text{red book/röd bok} & 0.053 \\ A \rightarrow \text{she/hon} & 0.053 \\ A \rightarrow \text{biology book/biologibok} & 0.053 \\ A \rightarrow \text{has/har} & 0.158 \\ A \rightarrow \text{it/det} & 0.053 \\ A \rightarrow \text{begun/börjat} & 0.053 \\ A \rightarrow \text{a/en} & 0.105 \end{array}$$

There are no shared bisegments left in the grammar now, so no more segmentations can be done. Obviously, the probability of the data given this new grammar is much smaller, but the grammar itself has generalized far beyond the training data, to the point where it largely agrees with the proposed trees in Figure 1 (except that this grammar binarizes the constituents, and treats *red book*/*röd bok* as a segment).

7 Conclusions

We have shown that there are situations in which a top-down segmenting approach that relies solely on binary segmentation will fail to generalize, despite there being ample evidence to a human that a generalization is warranted. We have proposed ternary segmentation as a solution to provide hypotheses that are considered good under a minimum description length objective. And we have shown that the proposed method could indeed perform generalizations that are clear to the human eye, but not discoverable through binary segmentation. The algorithm is comparable to previous segmentation approaches in terms of time and space complexity, so scaling up to non-toy training corpora is likely to work when the time comes.

Acknowledgements

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) under BOLT contract nos. HR0011-12-C-0014 and HR0011-12-C-0016, and GALE contract nos. HR0011-06-C-0022 and HR0011-06-C-0023; by the European Union under the FP7 grant agreement no. 287658; and by the Hong Kong Research Grants Council (RGC) research grants GRF620811, GRF621008, and GRF612806. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, the EU, or RGC.

References

Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation, and Compiling*. Prentice-Halll, Englewood Cliffs, New Jersey, 1972.

Phil Blunsom, Trevor Cohn, Chris Dyer, and Miles Osborne. A Gibbs sampler for phrasal synchronous grammar induction. In *Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP 2009)*, pp. 782–790, Suntec, Singapore, August 2009.

David Chiang. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228, 2007.

Frans Josef Och, Christoph Tillmann, and Hermann Ney. Improved alignment models for statistical machine translation. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pp. 20–28, University of Maryland, College Park, Maryland, June 1999.

Markus Saers and Dekai Wu. Bayesian induction of bracketing inversion transduction grammars. In *Sixth International Joint Conference on Natural Language Processing (IJCNLP2013)*, pp. 1158–1166, Nagoya, Japan, October 2013. Asian Federation of Natural Language Processing.

Markus Saers, Joakim Nivre, and Dekai Wu. Learning stochastic bracketing inversion transduction grammars with a cubic time biparsing algorithm. In *11th International Conference on Parsing Technologies (IWPT'09)*, pp. 29–32, Paris, France, October 2009.

Markus Saers, Karteek Addanki, and Dekai Wu. From finite-state to inversion transductions: Toward unsupervised bilingual grammar induction. In *24th International Conference on Computational Linguistics (COLING 2012)*, pp. 2325–2340, Mumbai, India, December 2012.

Markus Saers, Karteek Addanki, and Dekai Wu. Augmenting a bottom-up ITG with top-down rules by minimizing conditional description length. In *Recent Advances in Natural Language Processing (RANLP 2013)*, Hissar, Bulgaria, September 2013.

Markus Saers, Karteek Addanki, and Dekai Wu. Combining top-down and bottom-up search for unsupervised induction of transduction grammars. In *Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-7)*, pp. 48–57, Atlanta, Georgia, June 2013.

Markus Saers, Karteek Addanki, and Dekai Wu. Iterative rule segmentation under minimum description length for unsupervised transduction grammar induction. In Adrian-Horia Dediu, Carlos Martín-Vide, Ruslan Mitkov, and Bianca Truthe, editors, *Statistical Language and Speech Processing, First International Conference, SLSP 2013*, Lecture Notes in Artificial Intelligence (LNAI). Springer, Tarragona, Spain, July 2013.

Markus Saers, Karteek Addanki, and Dekai Wu. Unsupervised transduction grammar induction via minimum description length. In *Second Workshop on Hybrid Approaches to Translation (HyTra)*, pp. 67–73, Sofia, Bulgaria, August 2013.

Markus Saers. *Translation as Linear Transduction: Models and Algorithms for Efficient Learning in Statistical Machine Translation*. PhD thesis, Uppsala University, Department of Linguistics and Philology, 2011.

Dekai Wu. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403, 1997.

Hao Zhang, Chris Quirk, Robert C. Moore, and Daniel Gildea. Bayesian learning of non-compositional phrases with synchronous parsing. In *46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-08: HLT)*, pp. 97–105, Columbus, Ohio, June 2008.