Approximating Context-Free by Rational Transduction for Example-Based MT

Mark-Jan Nederhof*
AT&T Labs-Research, 180 Park Avenue, Florham Park, NJ 07932
and
Alfa Informatica (RUG), P.O. Box 716, NL-9700 AS Groningen, The Netherlands

Abstract

Existing studies show that a weighted context-free transduction of reasonable quality can be effectively learned from examples. This paper investigates the approximation of such transduction by means of weighted *rational* transduction. The advantage is increased processing speed, which benefits real-time applications involving spoken language.

1 Introduction

Several studies have investigated automatic or partly automatic learning of transductions for machine translation. Some of these studies have concentrated on finite-state or extended finite-state machinery, such as (Vilar and others, 1999), others have chosen models closer to context-free grammars and context-free transduction, such as (Alshawi et al., 2000; Watanabe et al., 2000; Yamamoto and Matsumoto, 2000), and yet other studies cannot be comfortably assigned to either of these two frameworks, such as (Brown and others, 1990) and (Tillmann and Ney, 2000).

In this paper we will investigate both context-free and finite-state models. The basis for our study is context-free transduction since that is a powerful model of translation, which can in many cases adequately describe the changes of word order between two languages, and the selection of appropriate lexical items. Furthermore, for limited domains, automatic learning of weighted context-free transductions from examples seems to be reasonably successful.

However, practical algorithms for computing the most likely context-free derivation have a cubic time complexity, in terms of the length of the input string, or in the case of a graph output by a speech recognizer, in terms of the number of nodes in the graph. For certain lexicalized context-free models we even obtain higher time complexities when the size of the grammar is not to be considered as a parameter (Eisner and Satta, 1999). This may pose problems, especially for real-time speech systems.

Therefore, we have investigated approximation of weighted context-free transduction by means of weighted rational transduction. The finite-state machinery for implementing the latter kind of transduction in general allows faster processing. We can also more easily obtain robustness. We hope the approximating model is able to preserve some of the accuracy of the context-free model.

In the next section, we discuss preliminary definitions, adapted from existing literature, making no more than small changes in presentation. In Section 3 we explain how context-free transduction grammars can be represented by ordinary context-free grammars, plus a phase of postprocessing. The approximation is discussed in Section 4. As shown in Section 5, we may easily process input in a robust way, ensuring we always obtain output. Section 6 discusses empirical results, and we end the paper with conclusions.

*The second address is the current contact address; supported by the Royal Netherlands Academy of Arts and Sciences; current secondary affiliation is the German Research Center for Artificial Intelligence (DFKI).

2 Preliminaries

2.1 hierarchical alignment

The input to our algorithm is a corpus consisting of pairs of sentences related by an *hierarchical alignment* (Alshawi et al., 2000). In what follows, the formalization of this concept has been slightly changed with respect to the above reference, to suit our purposes in the remainder of this article.

The hierarchically aligned sentence pairs in the corpus are 5-tuples $(v_1,\ v_2,\ f_1,\ f_2, r)$ satisfying the following. The first two components, v_1 and v_2, are strings, called the *source* string and the *target* string, respectively, the lengths of which are denoted by $n_1 = |v_1|$ and $n_2 = |v_2|$. We let $\mathcal{N}_1$ and $\mathcal{N}_2$ denote the sets of *string positions* $\{1, \ldots, n_1\}$ and $\{1, \ldots, n_2\}$ respectively.

Further, f_1 (resp. f_2) is a mapping from positions in $\mathcal{N}_1 \cup \{0\}$ (resp. $\mathcal{N}_2 \cup \{0\}$) to pairs of lists of positions from $\mathcal{N}_1$ (resp. $\mathcal{N}_2$), satisfying the following: if a position i is mapped to a pair (l_1, l_2), then the positions in the list $l_1 \cdot [i] \cdot l_2$ are in strictly increasing order; we let "$\cdot$" denote list-concatenation, and $[i]$ represents a list consisting of a single element i.

Each position in $\mathcal{N}_1$ (resp. $\mathcal{N}_2$) should occur at most once in the image of f_1 (resp. f_2). This means that f_1 and f_2 assign dependency structures to the source and target strings.

A further restriction on f_1 and f_2 requires some auxiliary definitions. Let f be either f_1 or f_2. We define $\hat{f}$ as the function that maps each position i to the list of positions $\hat{f}(j_1) \cdot \ldots \cdot \hat{f}(j_{m_1}) \cdot [i] \cdot \hat{f}(h_1) \cdot \ldots \cdot \hat{f}(h_{m_2})$ when $f(i) = ([j_1, \ldots, j_{m_1}], [h_1, \ldots, h_{m_2}])$. If v is a string $a_1 \cdots a_n$, and l is a list $[j_1, \ldots, j_m]$ of string positions in v, then $v\#l$ represents the string $a_{j_1} \cdots a_{j_m}$. If i is a single position, then $v\#i$ represents the symbol a_i.

We now say that f is *projective* if $\hat{f}$ maps each position i to some interval of positions $[p, p + 1, \cdots, q - 1, q]$. We will assume that both f_1 and f_2 are projective. (Strictly speaking, our algorithm would still be applicable if they were not projective, but it would treat the hierarchical alignment as if the symbols in the source and target strings had been reordered to make f_1 and f_2 projective.) Furthermore, a reasonable hierarchical alignment satisfies $\hat{f}(0) = [0, 1, \ldots, n]$, where $n = n_1$ or $n = n_2$ when $f = f_1$ or $f = f_2$, respectively, which means that all symbols in the string are indirectly linked to the 'dummy' position 0.

Lastly, r is the union of $\{(0, 0)\}$ and a subset of $\mathcal{N}_1 \times \mathcal{N}_2$ that relates positions in the two strings. It is such that $(i_1, j), (i_2, j) \in r$ imply $i_1 = i_2$ and $(i, j_1), (i, j_2) \in r$ imply $j_1 = j_2$; in other words, a position in one string is related to at most one position in the other. Furthermore, for each $(i, j) \in r - \{(0, 0)\}$ there is a pair $(i', j') \in r$ such that i occurs in one of the two lists of $f_1(i')$ and j occurs in one of the two lists of $f_2(j')$; this means that positions can only be related if their respective "mother" positions are related.

Note that this paper does *not* discuss how hierarchical alignments can be obtained from unannotated corpora of bitexts. This is the subject of existing studies, such as (Alshawi et al., 2000).

2.2 context-free transduction

Context-free transduction was originally called *syntax-directed transduction* in (Lewis II and Stearns, 1968), but since in modern formal language theory and computational linguistics the term "syntax" has a much wider range of meanings than just "context-free syntax", we will not use the original term here.

A (context-free) transduction grammar is a 5-tuple $(N, \Sigma_1, \Sigma_2, P, S)$, where N is a finite set of nonterminals, $S \in N$ is the start symbol, Σ_1 and Σ_2 are the source and target alphabets, and P is a finite set of productions of the form $A \rightarrow (\alpha, \beta)$, where $A \in N$, $\alpha \in (N \cup \Sigma_1)^*$ and $\beta \in (N \cup \Sigma_2)^*$, such that each nonterminal in α occurs exactly once in β and each nonterminal in β occurs exactly once in α.[1]

If we were to replace each RHS pair by only its first part α, we would obtain a context-free grammar for the source language, and if we were to replace each RHS pair by its second part β, we would obtain a context-free grammar for the target language. The combination of the two halves of such a RHS indicates how a parse for

[1] Note that we ignore the case that a single nonterminal occurs twice or more in α or β; if we were to include this case, some tedious complications of notation would result, without any theoretical gain such as an increase of generative power. We refer to (Lewis II and Stearns, 1968) for the general case.

the source language can be related to a parse for the target language, and this defines a transduction between the languages in an obvious way.

An example of a transduction grammar is:

$$\begin{aligned} \text{S} &\rightarrow (\textbf{Subj-IObj ``like'' Obj-Subj}, \\ &\quad\ \textbf{Obj-Subj Subj-IObj ``plaît''}) \\ \textbf{Subj-IObj} &\rightarrow (\textbf{``I''}, \textbf{``me''}) \\ \textbf{Obj-Subj} &\rightarrow (\textbf{``him''}, \textbf{``il''}) \end{aligned}$$

This transduction defines that a sentence **"I like him"** can be translated by **"il me plaît"**.

We can reduce the generative power of context-free transduction grammars by a syntactic restriction that corresponds to the bilexical context-free grammars (Eisner and Satta, 1999). Let us define a bilexical transduction grammar as a transduction grammar which is such that:

- there is a mapping from the set of nonterminals to $\Sigma_1 \times \Sigma_2$. Due to this property, we may write each nonterminal as $A[a, b]$ to indicate that it is mapped to the pair (a, b), where $a \in \Sigma_1$ and $b \in \Sigma_2$, where A is a so called *delexicalized* nonterminal. We may write S as $A[\vdash, \vdash]$, where $\vdash$ is a dummy symbol at the dummy string position 0. Further,

- each production is of one of the following five forms:

$$\begin{aligned} A[a,b] &\rightarrow (B[a,b]C[c,d],\ B[a,b]C[c,d]) \\ A[a,b] &\rightarrow (B[a,b]C[c,d],\ C[c,d]B[a,b]) \\ A[a,b] &\rightarrow (C[c,d]B[a,b],\ B[a,b]C[c,d]) \\ A[a,b] &\rightarrow (C[c,d]B[a,b],\ C[c,d]B[a,b]) \\ A[a,b] &\rightarrow (a,b) \end{aligned}$$

For convenience, we also allow productions of the form:

$$A[a,b] \quad \rightarrow \quad (x_1\, B[a,b]\, y_1,\ x_2\, B[a,b]\, y_2)$$

where $x_1, y_1 \in \Sigma_1^*$ and $x_2, y_2 \in \Sigma_2^*$.

In the experiments in Section 6, we also consider nonterminals that are lexicalized only by the source alphabet, which means that these nonterminals can be written as $A[a]$, where $a \in \Sigma_1$. The motivation is to restrict the grammar size and to increase the coverage.

Bilexical transduction grammars are equivalent to the dependency transduction model from (Alshawi et al., 2000).

2.3 obtaining a context-free transduction from the corpus

We extract a context-free transduction grammar from a corpus of hierarchical alignments, by locally translating each hierarchical alignment into a set of productions. The union of all these sets for the whole corpus is then the transduction grammar. Counting the number of times that identical productions are generated allows us to assign probabilities to the productions by maximum likelihood estimation.

We will consider a method that uses only one delexicalized nonterminal A. For a pair $(i, i') \in r$, we have a nonterminal $A[v_1\#i, v_2\#i']$ or a nonterminal $A[v_1\#i]$, depending on whether nonterminals are lexicalized by both source and target alphabets, or by just the source alphabet. Let us call that nonterminal $Nont(i, i')$.

Each pair of positions $(i, i') \in r$ gives rise to one production. Suppose that

$$f_1(i) = ([j_1, \ldots, j_{m_1}], [h_1, \ldots, h_{m_2}])$$

and each position in this pair is related by r to some position from $\mathcal{N}_2$, which we will call $j'_1, \ldots, j'_{m_1}, h'_1, \ldots, h'_{m_2}$, respectively, and similarly, suppose that

$$f_2(i') = ([j''_1, \ldots, j''_{m_3}], [h''_1, \ldots, h''_{m_4}])$$

and each position in this pair is related by r to some position from $\mathcal{N}_1$, which we will call $j'''_1, \ldots, j'''_{m_3}, h'''_1, \ldots, h'''_{m_4}$. Then the production is given by

$$\begin{aligned} &Nont(i,i') \ \rightarrow \\ &\quad (Nont(j_1, j'_1)\ \cdots\ Nont(j_{m_1}, j'_{m_1})\ v_1\#i \\ &\quad\ Nont(h_1, h'_1)\ \cdots\ Nont(h_{m_2}, h'_{m_2}), \\ &\quad Nont(j''_1, j'''_1)\ \cdots\ Nont(j''_{m_3}, j'''_{m_3})\ v_2\#i' \\ &\quad\ Nont(h''_1, h'''_1)\ \cdots\ Nont(h''_{m_4}, h'''_{m_4})\ ) \end{aligned}$$

Note that both halves of the RHS contain the same nonterminals but possibly in a different order.

However, if any position in $f_1(i)$ or $f_2(i')$ is not related to some other position by r, then the production above contains, instead of a nonterminal, a substring on which that position is projected by $\hat{f}_1$ or $\hat{f}_2$, respectively. E.g. if there is no position j'_1 such that $(j_1, j'_1) \in r$, then instead of $Nont(j_1, j'_1)$ we have the string $v_1\#\hat{f}_1(j_1)$.

In general, we cannot adapt the above algorithm to produce transduction grammars that are bilexical. For example, a production of the form:

$$A[a, a'] \rightarrow (A[b, b'] \; A[c, c'] \; a, \; A[c, c'] \; A[b, b'] \; a')$$

cannot be broken up into smaller, bilexical productions.[2] However, the hierarchical alignments that we work with were produced by an algorithm that ensures that bilexical grammars suffice. Formally, this applies when the following cannot occur: there are $i, i_1, i_2 \in \mathcal{N}_1$ and $j, j_1, j_2 \in \mathcal{N}_2$ such that $(i, j) \in r$, i_1 and i_2 occur in $f(i)$, j_1 and j_2 occur in $f(j)$ and $(i_1, j_1), (i_2, j_2) \in r$, and either $i_1 < i_2 < i$ and $j_2 < j_1 < j$, or $i < i_1 < i_2$ and $j < j_2 < j_1$, or $i_1 < i_2 < i$ and $j < j_1 < j_2$, or $i < i_1 < i_2$ and $j_1 < j_2 < j$.

For example, if the non-bilexical production we would obtain is:

$$A[a, a'] \rightarrow (A[b, b'] \; d \; a \; A[c, c'], \\ A[c, c'] \; e \; A[b, b'] \; a')$$

then the bilexical transduction grammar that our algorithm produces contains:

$$\begin{aligned}
A[a, a'] &\rightarrow (A[a, a'] \; A[c, c'], \; A[c, c'] \; A[a, a']) \\
A[a, a'] &\rightarrow (A[a, a'], \; e \; A[a, a']) \\
A[a, a'] &\rightarrow (A[b, b'] \; A[a, a'], \; A[b, b'] \; A[a, a']) \\
A[a, a'] &\rightarrow (d \; A[a, a'], \; A[a, a']) \\
A[a, a'] &\rightarrow (a, \; a')
\end{aligned}$$

3 Reordering as postprocessing

In the following section we will discuss an algorithm that was devised for context-free grammars. To make it applicable to transduction, we propose a way to represent bilexical transduction grammars as ordinary context-free grammars. In the new productions, symbols from the source and target alphabets occur side by side, but whereas source symbols are matched by the parser to the input, the target symbols are gathered into output strings. In our case, the unique output string the parser eventually produces from an input string is obtained from the most likely derivation that matches that input string.

[2]That bilexical transduction grammars are less powerful than arbitrary context-free transduction grammars can be shown formally; cf. Section 3.2.3 of (Aho and Ullman, 1972).

That the nonterminals in both halves of a RHS in the transduction grammar may occur in a different order is solved by introducing three special symbols, the *reorder operators*, which are interpreted after the parsing phase. These three operators will be written as "[", "|" and "]". In a given string, there should be matching triples of these operators, in such a way that if there are two such triples, then they either occur in two isolated substrings, or one occurs nested between the "[" and the "|" or nested between the "|" and the "]" of the other triple. The interpretation of an occurrence of a triple, say in an output string $v_1[v_2|v_3]v_4$, is that the two enclosed substrings should be reordered, so that we obtain $v_1 v_3 v_2 v_4$.

Both the reorder operators and the symbols of the target alphabet will here be marked by a horizontal line to distinguish them from the source alphabet. For example, the two productions

$$\begin{aligned}
A[a, a'] &\rightarrow (A[a, a'] \; A[c, c'], \; A[c, c'] \; A[a, a']) \\
A[a, a'] &\rightarrow (a, \; a')
\end{aligned}$$

from the transduction grammar are represented by the following two context-free productions:

$$\begin{aligned}
A[a, a'] &\rightarrow \overline{[} \; A[a, a'] \; \overline{|} \; A[c, c'] \; \overline{]} \\
A[a, a'] &\rightarrow a \; \overline{a'}
\end{aligned}$$

In the first production, the RHS nonterminals occur in the same order as in the left half of the original production, but reorder operators have been added to indicate that, after parsing, some substrings of the output string are to be reordered.

Our reorder operators are similar to the two operators $<$ and $>$ from (Vilar and others, 1999), but the former are more powerful, since the latter allow only single words to be moved instead of whole phrases.

4 Finite-state approximation

There are several methods to approximate context-free grammars by regular languages (Nederhof, 2000). We will consider here only the so called RTN method, which is applied in a simplified form.[3]

[3]As opposed to (Nederhof, 2000), we assume here that *all* nonterminals are mutually recursive, and the grammar contains self-embedding. We have observed that typical grammars that we obtain in the context of this article indeed have the property that *almost* all nonterminals belong to the same mutually recursive set.

A finite automaton is constructed as follows. For each nonterminal A from the grammar we introduce two states q_A and q'_A. For each production $A \rightarrow X_1 \cdots X_m$ we introduce $m+1$ states $q_0, \ldots, q_m$, and we add epsilon transitions from q_A to q_0 and from q_m to q'_A. The initial state of the automaton is q_S and the only final state is q'_S, where S is the start symbol of the grammar.

If a symbol X_i in the RHS of a production is a terminal, then we add a transition from q_{i-1} to q_i labelled by X_i. If a symbol X_i in the RHS is a nonterminal B, then we add epsilon transitions from q_{i-1} to q_B and from q'_B to q_i.

The resulting automaton is determinized and minimized to allow fast processing of input. Note that if we apply the approximation to the type of context-free grammar discussed in Section 3, the transitions include symbols from both source and target alphabets, but we treat both uniformly as input symbols for the purpose of determinizing and minimizing. This means that the driver for the finite automaton still encounters nondeterminism while processing an input string, since a state may have several outgoing transitions for different output symbols.

Furthermore, we ignore any weights that might be attached to the context-free productions, since determinization is problematic for weighted automata in general and in particular for the type of automaton that we would obtain when carrying over the weights from the context-free grammar onto the approximating language following (Mohri and Nederhof, 2001).

Instead, weights for the transitions of the finite automaton are obtained by training, using strings that are produced as a side effect of the computation of the grammar from the corpus. These strings contain the symbols from both the source and target strings mixed together, plus occurrences of the reorder operators where needed. A English/French example might be:

$$\overline{\lceil} \textbf{ I } \overline{\textbf{me}} \textbf{ like } \overline{\textbf{plaît}} \textbf{ } \overline{\rceil} \textbf{ him } \overline{\textbf{il}} \textbf{ } \overline{\rfloor}$$

The way these strings were obtained ensures that they are included in the language generated by the context-free grammar, and they are therefore also accepted by the approximating automaton due to properties of the RTN approximation. The weights are the negative log of the probabilities obtained by maximum likelihood estimation.

5 Robustness

The approximating finite automaton cannot ensure that the reorder operators "⌈", "|" and "⌉" occur in matching triples in output strings. There are two possible ways to deal with this problem. First, we could extend the driver of the finite automaton to only consider derivations in which the operators are matched. This is however counter to our need for very efficient processing, since we are not aware of any practical algorithms for finding matching brackets in paths in a graph of which the complexity is less than cubic.

Therefore, we have chosen a second approach, viz. to make the postprocessing robust, by inserting missing occurrences of "⌈" or "⌉" and removing redundant occurrences of brackets. This means that any string containing symbols from the target alphabet and occurrences of the reorder operators is turned into a string without reorder operators, with a change of word order where necessary.

Both the transduction grammar and, to a lesser extent, the approximating finite automaton suffer from not being able to handle all strings of symbols from the source alphabet. With finite-state processing however, it is rather easy to obtain robustness, by making the following three provisions:

1. To the nondeterministic finite automaton we add one epsilon transition from the initial state to q_A, for each nonterminal A. This means that from the initial state we may recognize an arbitrary phrase generated by some nonterminal from the grammar.

2. After the training phase of the weighted (minimal deterministic) automaton, all transitions that have not been visited obtain a fixed high (but finite) weight. This means that such transitions are only applied if all others fail.

3. The driver of the automaton is changed so that it restarts at the initial state when it gets stuck at some input word, and when necessary, that input word is deleted. The out-

put string with the lowest weight obtained so far (preferably attached to final states, or to other states with outgoing transitions labelled by input symbols) is then concatenated with the output string resulting from processing subsequent input.

6 Experiments

We have investigated a corpus of English/Japanese sentence pairs, related by hierarchical alignment (see also (Bangalore and Riccardi, 2001)). We have taken the first 500, 1000, 1500, ... aligned sentence pairs from this corpus to act as training corpora of varying sizes; we have taken 300 other sentence pairs to act as test corpus.

We have constructed a bilexical transduction grammar from each training corpus, in the form of a context-free grammar, and this grammar was approximated by a finite automaton. The input sentences from the test corpus were then processed by context-free and finite-state machinery (in the sequel referred to by **cfg** and **fa**, respectively). We have also carried out experiments with robust finite-state processing, as discussed in Section 5, which is referred to by **robust_fa**. If we append **2** after a tag, this mean that $Nont(i, i') = A[v_1 \# i, v_2 \# i']$, otherwise $Nont(i, i') = A[v_1 \# i]$ (see Section 2.3).

The reorder operators from the resulting output strings were applied in a robust way as explained in Section 5. The output strings were then compared to the reference output from the corpus, resulting in Figure 1. Our metric is word accuracy, which is based on edit distance. For a pair of strings, the edit distance is defined as the minimum number of substitutions, insertions and deletions needed to turn one string into the other. The word accuracy of a string v with regard to a string w is defined to be $1 - \frac{d}{n}$, where d is the edit distance between v and w and n is the length of w.

To allow a comparison with more established techniques (see e.g. (Bangalore and Riccardi, 2001)), we also take into consideration a simple bigram model, trained on the strings comprising both source and target sentences and reorder operators, as explained in Section 4. For the purposes of predicting output symbols, a series of consecutive target symbols and reorder operators following a source symbol in the training sentences are treated as a single symbol by the bigram model, and only those may be output after that source symbol. Since our construction is such that target symbols always *follow* source symbols they are a translation of (according to the automatically obtained hierarchical alignment), this modification to the bigram model prevents output of totally unrelated target symbols that could otherwise result from a standard bigram model. It also ensures that a bounded number of output symbols per input symbol are produced.

The fraction of sentences that were transduced (i.e. that were accepted by the grammar or the automaton), is indicated in Figure 2. Since **robust_fa(2)** and **bigram** are able to transduce all input, they are not represented here. Note that the average word accuracy is computed only with respect to the sentences that could be transduced, which explains the high accuracy for small training corpora in the cases of **cfg(2)** and **fa(2)**, where the few sentences that can be transduced are mostly short and simple.

Figure 3 presents the time consumption of transduction for the entire test corpus. These data support our concerns about the high costs of context-free processing, even though our parser relies heavily on lexicalization.[4]

Figure 4 shows the sizes of the automata after determinization and minimization. Determinization for the largest automata indicated in the Figure took more than 24 hours for both **fa(2)** and **robust_fa(2)** , which suggests these methods become unrealistic for training corpus sizes considerably larger than 10,000 bitexts.

7 Conclusions

For our application, context-free transduction has a relatively high accuracy, but it also has a high time consumption, and it may be difficult to obtain robustness without further increasing the time costs. These are two major obstacles for use in spoken language systems. We have tried to obtain a rational transduction that approximates a

[4] It uses a trie to represent productions (similar to ELR parsing (Nederhof, 1994)), postponing generation of output for a production until all nonterminals and all input symbols from the right-hand side have been found.

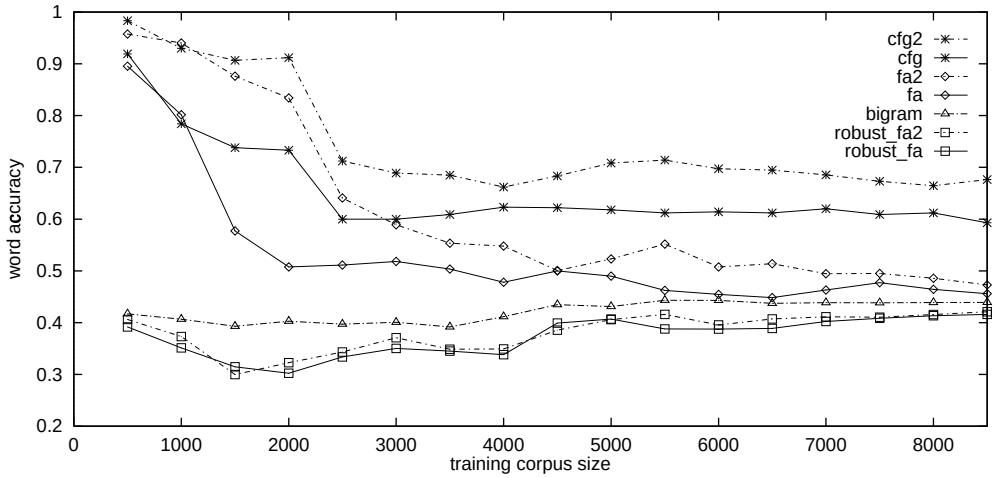

Figure 1: Average word accuracy for transduced sentences.

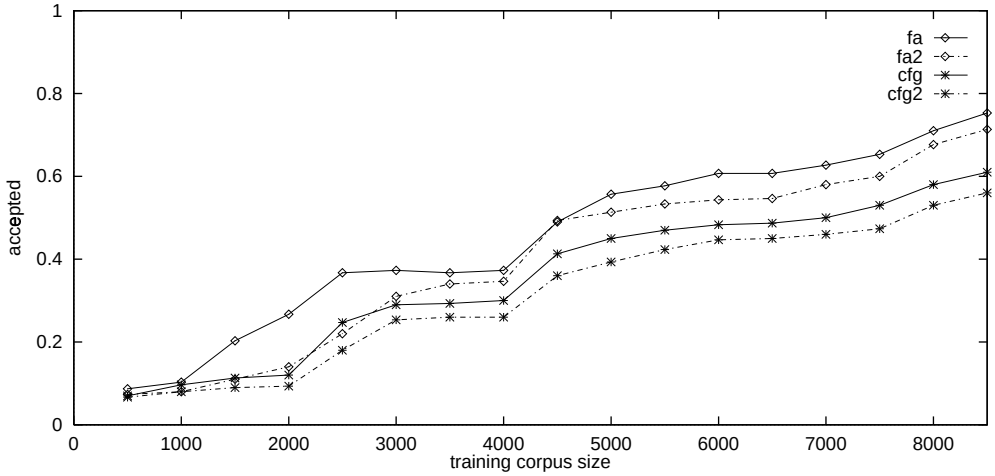

Figure 2: Fraction of the sentences that were transduced.

context-free transduction, preserving some of its accuracy.

Our experiments show that the automata we obtain become very large for training corpora of increasing sizes. This poses a problem for determinization. We conjecture that the main source of the excessive growth of the automata lies in noise in the bitexts and their hierarchical alignments. It is a subject for further study whether we can reduce the impact of this noise, e.g. by clustering of source symbols, or by removing some infrequent, idiosyncratic rules from the obtained transduction grammar. Also, other methods of regular approximation of context-free grammars may be considered.

In comparison to a simpler model, viz. bigrams, our approximating transductions do not have a very high accuracy, which is especially worrying since the off-line costs of computation are much higher than in the case of bigrams. The relatively low accuracy may be due to sparseness of data when attaching weights to transitions: the size of the minimal deterministic automaton grows much faster than the size of the training corpus it is constructed from, and the same training corpus is used to train the weights of the transitions of the automaton. Thereby, many transitions do not obtain accurate weights, and unseen input sentences are not translated accurately.

The problems described here may be avoided by leaving out the determinization of the automaton. This however leads to two new problems: training of the weights requires more sophisticated algorithms, and we may expect an increase in the time needed to transduce input sentences, since now both source and target symbols give

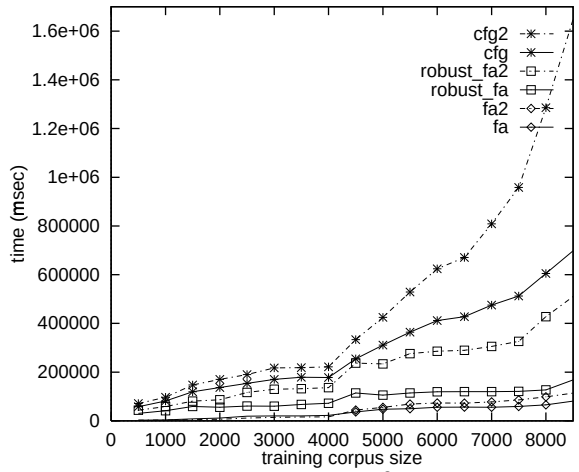

Figure 3: Time consumption of transduction.

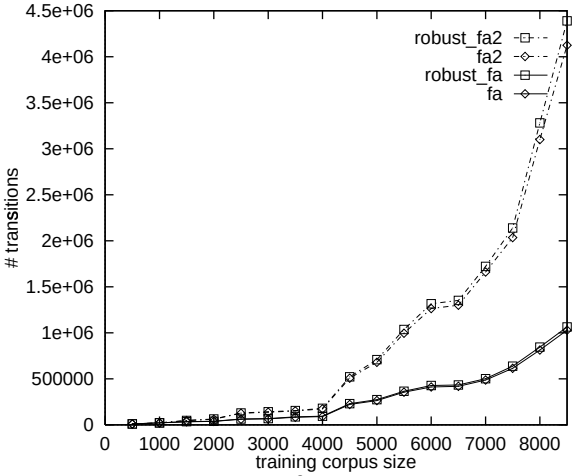

Figure 4: Sizes of the automata.

rise to nondeterminism. Whether these problems can be overcome requires further study.

Acknowledgements

This work is a continuation of partly unpublished experiments by Srinivas Bangalore, which includes regular approximation of grammars obtained from hierarchical alignments. Many ideas in this paper originate from frequent discussions with Hiyan Alshawi, Srinivas Bangalore and Mehryar Mohri, for which I am very grateful.

References

A.V. Aho and J.D. Ullman. 1972. *Parsing*, volume 1 of *The Theory of Parsing, Translation and Compiling*. Prentice-Hall.

H. Alshawi, S. Bangalore, and S. Douglas. 2000. Learning dependency translation models as collections of finite-state head transducers. *Computational Linguistics*, 26(1):45–60.

S. Bangalore and G. Riccardi. 2001. A finite-state approach to machine translation. In *2nd Meeting of the North American Chapter of the ACL*, Pittsburgh, PA, June.

P.F. Brown et al. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *37th Annual Meeting of the ACL*, pages 457–464, Maryland, June.

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

M. Mohri and M.-J. Nederhof. 2001. Regular approximation of context-free grammars through transformation. In J.-C. Junqua and G. van Noord, editors, *Robustness in Language and Speech Technology*, pages 153–163. Kluwer Academic Publishers.

M.-J. Nederhof. 1994. An optimal tabular parsing algorithm. In *32nd Annual Meeting of the ACL*, pages 117–124, Las Cruces, New Mexico, June.

M.-J. Nederhof. 2000. Practical experiments with regular approximation of context-free languages. *Computational Linguistics*, 26(1):17–44.

C. Tillmann and H. Ney. 2000. Word re-ordering and DP-based search in statistical machine translation. In *The 18th International Conference on Computational Linguistics*, pages 850–856, Saarbrücken, July–August.

J.M. Vilar et al. 1999. Text and speech translation by means of subsequential transducers. In A. Kornai, editor, *Extended finite state models of language*, pages 121–139. Cambridge University Press.

H. Watanabe, S. Kurohashi, and E. Aramaki. 2000. Finding structural correspondences from bilingual parsed corpus for corpus-based translation. In *The 18th International Conference on Computational Linguistics*, pages 906–912, Saarbrücken, July–August.

K. Yamamoto and Y. Matsumoto. 2000. Acquisition of phrase-level bilingual correspondence using dependency structure. In *The 18th International Conference on Computational Linguistics*, pages 933–939, Saarbrücken, July–August.