Chapter 11

Synchronous CFGs

Synchronous context-free grammars are a generalization of CFGs that generate pairs of related strings instead of single strings. They are useful in many situations where one might want to specify a recursive relationship between two languages. Originally, they were developed in the late 1960s for programming-language compilation (Aho and Ullman, 1969). In natural language processing, they have been used for machine translation (Wu, 1997; Yamada and Knight, 2001; Chiang, 2005) and (less commonly, perhaps) semantic interpretation.

The term *synchronous CFG* is recent and far from universal. They were originally known as *syntax-directed transduction grammars* (Lewis and Stearns, 1968) or *syntax-directed translation schemata* (Aho and Ullman, 1969). *Inversion transduction grammars* (Wu, 1997) are a special case of synchronous CFGs.

11.1 Motivation

Earlier we used finite transducers to perform many kinds of string transformations, but there are many kinds of transformations that they can't perform. Even something as simple as reversing the input string is beyond the power of finite transducers.

Question 18. Informally, what is the reason that there is no finite transducer that can output the reverse of its input string?

For a linguistic example, consider the following English sentence and its (admittedly somewhat unnatural) equivalent in Japanese (with English glosses):

(11.1) the boy stated that the student said that the teacher danced

shoonen-ga	gakusei-ga	sensei-ga	odotta	to	itta	to	hanasita
the boy	the student	the teacher	danced	that	said	that	stated

This kind of reordering is beyond the power of finite transducers, but a synchronous CFG can do this.

11.2 Definition

In a synchronous CFG, the productions have two right-hand sides—call them the *input rhs* and the *output rhs*—that are related in a certain way. Below is an example synchronous CFG for a fragment of English

and Japanese:

$$\text{S} \rightarrow \langle \text{NP}\boxed{1}\ \text{VP}\boxed{2}, \text{NP}\boxed{1}\ \text{VP}\boxed{2} \rangle \tag{11.2}$$

$$\text{VP} \rightarrow \langle \text{VB}\boxed{1}\ \text{NP}\boxed{2}, \text{NP}\boxed{2}\ \text{VB}\boxed{1} \rangle \tag{11.3}$$

$$\text{NP} \rightarrow \langle \text{I}, \text{watashi wa} \rangle \tag{11.4}$$

$$\text{NP} \rightarrow \langle \text{the box}, \text{hako wo} \rangle \tag{11.5}$$

$$\text{VB} \rightarrow \langle \text{open}, \text{akemasu} \rangle \tag{11.6}$$

The boxed numbers $\boxed{i}$ link up nonterminal symbols in the input rhs with nonterminal symbols in the output rhs: $\boxed{1}$ links to $\boxed{1}$, $\boxed{2}$ with $\boxed{2}$, and so on. Linked nonterminals must match (X with X), and the linking must be a one-to-one correspondence.

How does this grammar work? Just as we start in a CFG with a start symbol and repeatedly rewrite nonterminal symbols using the productions, so in a synchronous CFG, we start with a pair of linked start symbols (I just chose the number 10 arbitrarily),

$$\langle \text{S}\boxed{10}, \text{S}\boxed{10} \rangle$$

and repeatedly rewrite pairs of nonterminal symbols using the productions—with two wrinkles. First, when we apply a production, we renumber the boxed indices consistently to fresh indices that aren't in our working string pair. Thus, applying production (11.2), we get

$$\Rightarrow \langle \text{NP}\boxed{11}\ \text{VP}\boxed{12}, \text{NP}\boxed{11}\ \text{VP}\boxed{12} \rangle$$

Second, we are only allowed to rewrite *linked* nonterminal symbols. Thus we can apply production (11.3) like so:

$$\Rightarrow \langle \text{NP}\boxed{11}\ \text{VB}\boxed{13}\ \text{NP}\boxed{14}, \text{NP}\boxed{11}\ \text{NP}\boxed{14}\ \text{VB}\boxed{13} \rangle$$

But now if we want to apply production (11.4), we can't apply it to $\text{NP}\boxed{11}$ on one side and $\text{NP}\boxed{14}$ on the other, like this:

$$\not\Rightarrow \langle \text{I VB}\boxed{13}\ \text{NP}\boxed{14}, \text{NP}\boxed{11}\ \text{watashi wa VB}\boxed{13} \rangle$$

But we can apply it to any linked nonterminals, like so:

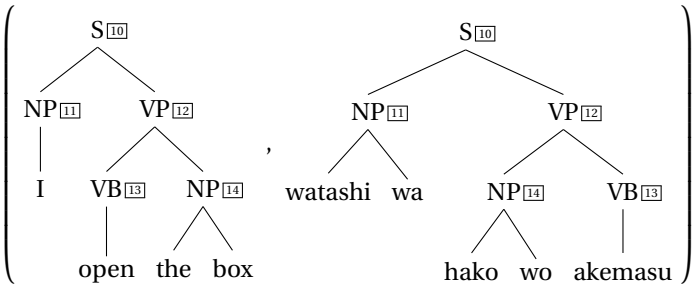
$$\Rightarrow \langle \text{I VB}\boxed{13}\ \text{NP}\boxed{14}, \text{watashi wa NP}\boxed{14}\ \text{VB}\boxed{13} \rangle$$

$$\Rightarrow \langle \text{I open NP}\boxed{14}, \text{watashi wa NP}\boxed{14}\ \text{akemasu} \rangle$$

$$\Rightarrow \langle \text{I open the box}, \text{watashi wa hako wo akemasu} \rangle$$

And now we have an English string and Japanese string which are translations of each other!

We can also view synchronous CFG derivations as pairs of trees, just as CFG derivations can be viewed as trees:



(11.7)

By this point, we often don't care about the boxed numbers and therefore drop them.

Here is a more complicated example, a grammar deriving sentence pair (11.1):

$$\begin{aligned}
\text{S} &\rightarrow \langle \text{NP}\boxed{1}\ \text{VP}\boxed{2}, \text{NP}\boxed{1}\ \text{VP}\boxed{2} \rangle \\
\text{VP} &\rightarrow \langle \text{VB}\boxed{1}, \text{VB}\boxed{1} \rangle \\
\text{VP} &\rightarrow \langle \text{VB}\boxed{1}\ \text{SBAR}\boxed{2}, \text{SBAR}\boxed{2}\ \text{VB}\boxed{1} \rangle \\
\text{SBAR} &\rightarrow \langle \text{IN}\boxed{1}\ \text{S}\boxed{2}, \text{S}\boxed{2}\ \text{IN}\boxed{1} \rangle \\
\text{IN} &\rightarrow \langle \text{that}, \text{to} \rangle \\
\text{NP} &\rightarrow \langle \text{the boy}, \text{shoonen-ga} \rangle \\
\text{NP} &\rightarrow \langle \text{the student}, \text{gakusei-ga} \rangle \\
\text{NP} &\rightarrow \langle \text{the teacher}, \text{sensei-ga} \rangle \\
\text{VB} &\rightarrow \langle \text{danced}, \text{odotta} \rangle \\
\text{VB} &\rightarrow \langle \text{said}, \text{itta} \rangle \\
\text{VB} &\rightarrow \langle \text{stated}, \text{hanasita} \rangle
\end{aligned} \tag{11.8}$$

Question 19. Show how to derive sentence pair (11.1).

Question 20. How would you write a synchronous CFG that outputs the reverse of its input string?

We can add states to a synchronous CFG just like we did for CFGs. We introduce the following notational convention: if β is a string of terminals and r indexed nonterminals, then $\beta[q_1, \ldots, q_r]$ adds state q_i to the nonterminal indexed by i.

11.3 Weighted synchronous CFGs

In a *weighted synchronous CFG*, a weight is attached to each production. The weight of a whole derivation is just the product of the weights of the productions used in the derivation. Thus a weighted synchronous CFG generates a weighted set of pairs of derivations.

We can (but don't necessarily have to) think of a weighted synchronous CFG as a stochastic process, in at least two different ways. First, by analogy with PCFG, each production $A \rightarrow \langle \alpha, \alpha' \rangle$ could have probability $P(\alpha, \alpha' \mid A)$, so that the probability of the whole derivation is the joint probability of the input and output trees. Or, by analogy with finite transducers, each production could have probability $P(\alpha' \mid A, \alpha)$, in which case the probability of the whole derivation would be the conditional probability of the output tree given the input tree.

For example, suppose we want to translate French into English. We could learn a weighted synchronous CFG that translates from English trees T_e into French trees T_f and computes $P(T_f \mid T_e)$. Then we would like to combine this grammar with a language model $P(T_e)$ so that we can compute:

$$\arg\max_{T_e} P(T_e \mid T_f) = \arg\max_{T_e} P(T_f, T_e) \tag{11.9}$$

$$= \arg\max_{T_e} P(T_f \mid T_e) P(T_e) \tag{11.10}$$

But suppose we are given only a French string, not a French tree, as input, and want only an English

string, not an English tree, as output. Then:

$$\arg\max_e P(e \mid f) = \arg\max_e P(f, e) \tag{11.11}$$

$$= \arg\max_e \sum_{T_f} \sum_{T_e} P(T_f, T_e) \tag{11.12}$$

$$= \arg\max_e \sum_{T_f} \sum_{T_e} P(T_f \mid T_e) P(T_e) \tag{11.13}$$

The summations are over trees that yield f and e, respectively. It is tractable to compute both summations for a fixed f, e, but maximizing over e is intractable. Therefore the standard practice is to use the **Viterbi approximation**:

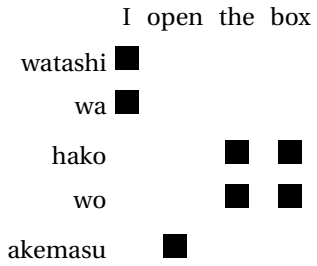
$$= \arg\max_{T_f, T_e} P(T_f \mid T_e) P(T_e) \tag{11.14}$$

where again the maximization over T_f and T_e is over trees that yield f and e, respectively. We will see how to do this maximization below.

This $P(T_e)$ could be modeled by a PCFG. However, n-gram language models continue to be far more effective than tree language models. So it is common to use an n-gram language model instead, even though it is a string language model, and assume that $P(T_e) \approx P(e)$.

11.4 Extracting synchronous CFGs from data

There a few ways of getting a synchronous CFG from data (as opposed to writing one by hand). The simplest (in some sense) is a two-phase process. In the first phase, we use a word-alignment model (like IBM Model 1/2 or the HMM word alignment model) to produce a word alignment, which we can visualize like this:



In the second phase, we extract *all* possible synchronous CFG rules, using the nonterminal symbol X, that possibly could have been used to generate this sentence, subject to the constraint that if a word appears in a rule, then all the words it's aligned with also appear in the rule. For example, starting with rules that have no nonterminals on the rhs, all of the following would be rules:

$X \rightarrow$ (watashi wa, I)

$X \rightarrow$ (hako wo, the box)

$X \rightarrow$ (akemasu, open)

$X \rightarrow$ (hako wo akemasu, open the box)

$X \rightarrow$ (watashi wa hako wo akemasu, I open the box)

We can also "subtract" one rule from another to get rules with rhs nonterminals:

$$X \to (X\boxed{1} \text{ akemasu}, \text{open } X\boxed{1})$$
$$X \to (\text{hako wo } X\boxed{1}, X\boxed{1} \text{ the box})$$
$$X \to (X\boxed{1}\ X\boxed{2}, X\boxed{2}\ X\boxed{1})$$
$$X \to (X\boxed{1} \text{ hako wo akemasu}, X\boxed{1} \text{ open the box})$$
$$X \to (\text{watashi wa } X\boxed{1} \text{ akemasu}, \text{I open } X\boxed{1})$$
$$X \to (\text{watashi wa hako wo } X\boxed{1}, \text{I } X\boxed{1} \text{ the box})$$
$$X \to (X\boxed{1}\ X\boxed{2} \text{ akemasu}, X\boxed{1} \text{ open } X\boxed{2})$$
$$X \to (X\boxed{1} \text{ hako wo } X\boxed{2}, X\boxed{1}\ X\boxed{2} \text{ the box})$$
$$X \to (\text{watashi wa } X\boxed{1}\ X\boxed{2}, \text{I } X\boxed{2}\ X\boxed{1})$$
$$X \to (\text{watashi wa } X\boxed{1}, \text{I } X\boxed{1})$$
$$X \to (X\boxed{1}\ X\boxed{2}\ X\boxed{3}, X\boxed{1}\ X\boxed{3}\ X\boxed{2})$$

The number of rules thus extracted is exponential in the sentence length, so invariably some constraints are imposed, like:

- A rule can have at most two nonterminals in each rhs
- A rule can have at most (say) 10 terminals plus nonterminals in each rhs
- A rule must have at least one aligned word pair

Even under such restrictions, the total number of rules can exceed a billion for large training datasets!

Estimating the probabilities of these rules is a messy business, because the rules overlap, so we have no way of determining how many times each rule occurred. The simplest method (in my opinion) is

$$p(X \to (\gamma, \alpha)) = \frac{c(X \to (\gamma, \alpha))}{\sum_{\gamma', \alpha'} c(X \to (\gamma', \alpha'))}, \tag{11.15}$$

where $c(X \to (\gamma, \alpha))$ is the number of *sentences* in which rule $(X \to (\gamma, \alpha))$ was found (in other words, we don't count within-sentence duplicates).

It's also extremely common to impose syntactic constraints. If we have a parse tree for the Japanese side or the English side, then we can require that every lhs or rhs nonterminal corresponds to a node of the tree. As a bonus, we can use the node's label as the nonterminal symbol instead of X.

11.5 Binarization

Define the *rank* of a right-hand side to be the number of nonterminals in it: for example, NP VP has rank two. Now define the rank of a CFG or synchronous CFG to be the maximum rank of any of its right-hand sides.

Recall that any (non-synchronous) CFG can be converted into a (weakly) equivalent CFG with rank two or less (Chomsky normal form). Is this true for synchronous CFG? It turns out that any synchronous CFG of rank three can be converted into a synchronous CFG of rank two. For example, the production

$$A \to \langle B\boxed{1}\ C\boxed{2}\ D\boxed{3}, D\boxed{3}\ B\boxed{1}\ C\boxed{2} \rangle$$

can be binarized into

$$A \rightarrow \langle A'\boxed{4}\ D\boxed{3}, D\boxed{3}\ A'\boxed{4}\rangle$$
$$A' \rightarrow \langle B\boxed{1}\ C\boxed{2}, B\boxed{1}\ C\boxed{2}\rangle$$

Note that we did not sever any links in doing so.

Question 21. Show that any synchronous CFG production of rank three can be converted into a set of productions of rank two. Assume that the production doesn't have any terminal symbols.

But there are synchronous CFGs of rank four that can't be binarized—namely, any synchronous CFG containing a production with one of the following forms:

$$\begin{aligned} A &\rightarrow \langle B\boxed{1}\ C\boxed{2}\ D\boxed{3}\ E\boxed{4}, D\boxed{3}\ B\boxed{1}\ E\boxed{4}\ C\boxed{2}\rangle \\ A &\rightarrow \langle B\boxed{1}\ C\boxed{2}\ D\boxed{3}\ E\boxed{4}, C\boxed{2}\ E\boxed{4}\ B\boxed{1}\ D\boxed{3}\rangle \end{aligned} \tag{11.16}$$

Question 22. Show that the first form can't be binarized in such a way that preserves all the links.

In general, let r-SCFG stand for the set of string relations generated by synchronous CFGs of rank r. Then:

$$1\text{-SCFG} \subsetneq 2\text{-SCFG} = 3\text{-SCFG} \subsetneq 4\text{-SCFG} \subsetneq \ldots \tag{11.17}$$

despite the fact that non-synchronous CFGs of rank 2 and higher are all weakly equivalent (Aho and Ullman, 1969). There is an efficient algorithm for minimizing the rank of a synchronous CFG (Zhang and Gildea, 2007).

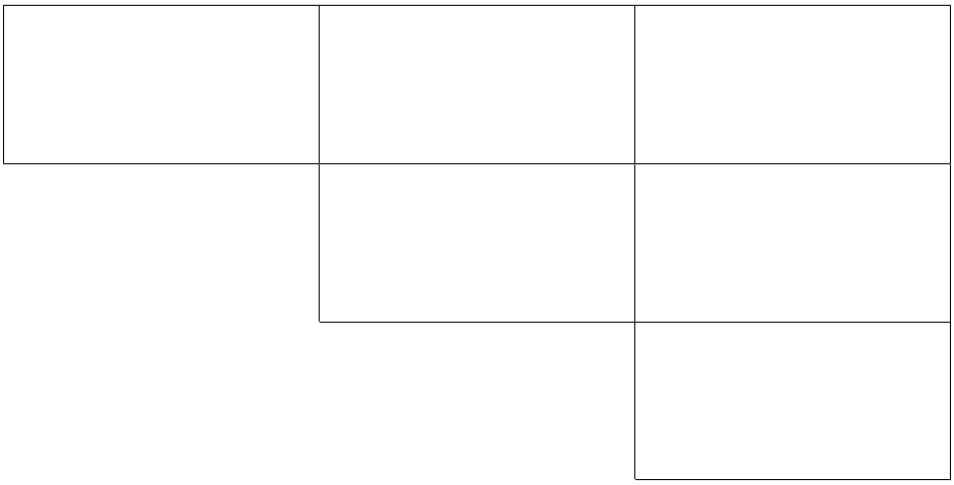
11.6 Translation

Translation, analogous to application of finite transducers, is the problem of finding all possible output strings for a given input string. If the input side of the grammar is in Chomsky normal form, we can use a simple variant of CKY to do translation.

Consider the following synchronous CFG:

$$\begin{aligned} \text{S} &\xrightarrow{1} \langle \text{NP}\boxed{1}\ \text{VP}\boxed{2}, \text{NP}\boxed{1}\ \text{VP}\boxed{2}\rangle \\ \text{VP} &\xrightarrow{1} \langle \text{VB}\boxed{1}\ \text{NP}\boxed{2}, \text{NP}\boxed{2}\ \text{VB}\boxed{1}\rangle \\ \text{VB} &\xrightarrow{1} \langle \text{see}, \text{veo}\rangle \\ \text{VB} &\xrightarrow{1} \langle \text{love}, \text{amo}\rangle \\ \text{NP} &\xrightarrow{0.3} \langle \text{I}, \text{yo}\rangle \\ \text{NP} &\xrightarrow{0.7} \langle \text{I}, \epsilon\rangle \\ \text{NP} &\xrightarrow{0.1} \langle \text{you}, \text{te}\rangle \\ \text{NP} &\xrightarrow{0.9} \langle \text{you}, \text{la}\rangle \\ \text{NP} &\xrightarrow{1} \langle \text{her}, \text{la}\rangle \end{aligned} \tag{11.18}$$

Question 23. Run the Viterbi CKY algorithm (just probabilities, no back-pointers) using the *input* side of the grammar on the input string *I see her*.



Remember that in CFG parsing, we represented back-pointers as CFG rules, augmenting nonterminal symbols with spans i, j to specify which cells a pointer points to. Similarly, in synchronous CFG translation, we represent back-pointers as synchronous CFG rules.

Question 24. Using the same grammar and input string as before, write down the back-pointers for the two missing cells:

NP : $\text{NP}_{0,1} \rightarrow \langle \text{I}, \epsilon \rangle$	$\emptyset$	
	VB : $\text{VB}_{1,2} \rightarrow \langle \text{see}, \text{veo} \rangle$	VP : $\text{VP}_{1,3} \rightarrow \langle \text{VB}_{1,2}\ \boxed{1}\ \text{NP}_{2,3}\ \boxed{2}, \text{NP}_{2,3}\ \boxed{2}\ \text{VB}_{1,2}\ \boxed{1} \rangle$

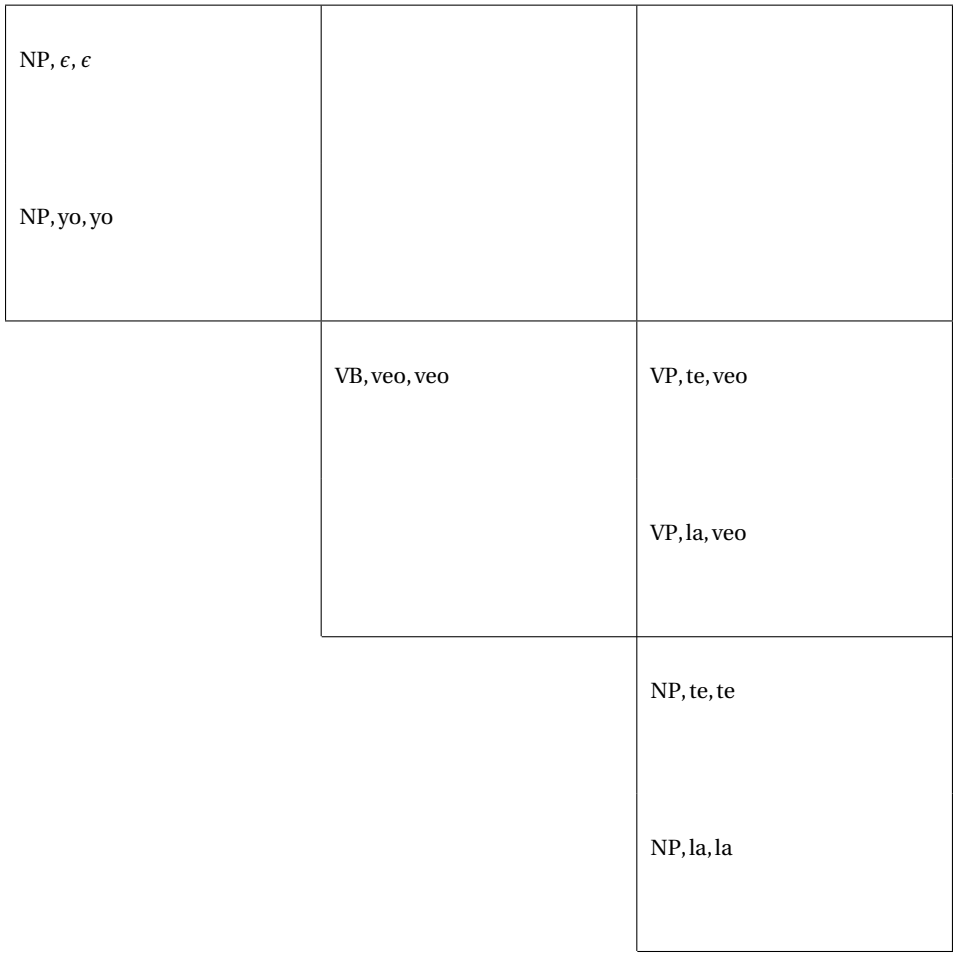
What is the best translation?

11.7 Translation with a language model

Suppose we want to do translation with a weighted synchronous CFG but also want to use a m-gram language model, such that the score of a derivation is the score according to the synchronous CFG multiplied by the language model score.

Recall that a m-gram language model can be thought of as a finite transducer, one that reads in a string, and outputs the same string, assigning it a probability. We sort of want to compose the synchronous CFG that represents our translation model with the finite transducer that represents the language model. We could in fact do exactly this, but we haven't talked about how to intersect CFGs with finite automata. And, it wouldn't be the most efficient way to do it.

The way this is usually done in practice is an algorithm due to Wu (1996). Let's assume a bigram language model. In the CKY algorithm, we add an entry X to cell (i, j) when we've determined that, according to the grammar, we can rewrite an X into the substring $w_{i+1} \cdots w_j$. Now, we modify the algorithm so that we add entry (X, a, b) to cell (i, j) when we can rewrite an X into the source-side substring $f_{i+1} \cdots f_j$, and the first and last words of the target-side translation are a and b, respectively.



Question 25. Fill in the remaining cells, and fill in the probabilities for all the items, assuming the fol-

lowing language model:

$$
\begin{aligned}
p(\texttt{<s>}\ \text{yo}) &= 0.1 \\
p(\texttt{<s>}\ \text{te}) &= 0.4 \\
p(\texttt{<s>}\ \text{la}) &= 0.5 \\
p(\text{yo te}) &= 0.5 \\
p(\text{yo la}) &= 0.5 \\
p(\text{te veo}) &= 1 \\
p(\text{la veo}) &= 1 \\
p(\text{veo}\ \texttt{</s>}) &= 1
\end{aligned}
$$

Bibliography

Aho, A. V. and J. D. Ullman (1969). "Syntax Directed Translations and the Pushdown Assembler". In: *Journal of Computer and System Sciences* 3, pp. 37–56.

Chiang, David (2005). "A Hierarchical Phrase-Based Model for Statistical Machine Translation". In: *Proc. ACL*, pp. 263–270.

Lewis P. M., II and R. E. Stearns (1968). "Syntax-Directed Transduction". In: *Journal of the ACM* 15, pp. 465–488.

Wu, Dekai (1996). "A Polynomial-Time Algorithm for Statistical Machine Translation". In: *Proc. ACL*, pp. 152–158.

— (1997). "Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora". In: *Computational Linguistics* 23, pp. 377–404.

Yamada, Kenji and Kevin Knight (2001). "A Syntax-based Statistical Translation Model". In: *Proc. ACL*, pp. 523–530.

Zhang, Hao and Daniel Gildea (2007). "Factorization of Synchronous Context-Free Grammars in Linear Time". In: *Proceedings of SSST*, pp. 25–32.