Prefix Probability for Probabilistic Synchronous Context-Free Grammars

Mark-Jan Nederhof
School of Computer Science
University of St Andrews
North Haugh, St Andrews, Fife
KY16 9SX
United Kingdom
markjan.nederhof@googlemail.com

Giorgio Satta
Dept. of Information Engineering
University of Padua
via Gradenigo, 6/A
I-35131 Padova
Italy
satta@dei.unipd.it

Abstract

We present a method for the computation of prefix probabilities for synchronous context-free grammars. Our framework is fairly general and relies on the combination of a simple, novel grammar transformation and standard techniques to bring grammars into normal forms.

1 Introduction

Within the area of statistical machine translation, there has been a growing interest in so-called syntax-based translation models, that is, models that define mappings between languages through hierarchical sentence structures. Several such statistical models that have been investigated in the literature are based on synchronous rewriting or tree transduction. Probabilistic synchronous context-free grammars (PSCFGs) are one among the most popular examples of such models. PSCFGs subsume several syntax-based statistical translation models, as for instance the stochastic inversion transduction grammars of Wu (1997), the statistical model used by the Hiero system of Chiang (2007), and systems which extract rules from parsed text, as in Galley et al. (2004).

Despite the widespread usage of models related to PSCFGs, our theoretical understanding of this class is quite limited. In contrast to the closely related class of probabilistic context-free grammars, a syntax model for which several interesting mathematical and statistical properties have been investigated, as for instance by Chi (1999), many theoretical problems are still unsolved for the class of PSCFGs.

This paper considers a parsing problem that is well understood for probabilistic context-free grammars but that has never been investigated in the context of PSCFGs, viz. the computation of prefix probabilities. In the case of a probabilistic context-free grammar, this problem is defined as follows. We are asked to compute the probability that a sentence generated by our model starts with a prefix string v given as input. This quantity is defined as the (possibly infinite) sum of the probabilities of all strings of the form vw, for any string w over the alphabet of the model. This problem has been studied by Jelinek and Lafferty (1991) and by Stolcke (1995). Prefix probabilities can be used to compute probability distributions for the next word or part-of-speech. This has applications in incremental processing of text or speech from left to right; see again (Jelinek and Lafferty, 1991). Prefix probabilities can also be exploited in speech understanding systems to score partial hypotheses in beam search (Corazza et al., 1991).

This paper investigates the problem of computing prefix probabilities for PSCFGs. In this context, a pair of strings v_1 and v_2 is given as input, and we are asked to compute the probability that any string in the source language starting with prefix v_1 is translated into any string in the target language starting with prefix v_2. This probability is more precisely defined as the sum of the probabilities of translation pairs of the form $[v_1w_1, v_2w_2]$, for any strings w_1 and w_2.

A special case of prefix probability for PSCFGs is the right prefix probability. This is defined as the probability that some (complete) input string w in the source language is translated into a string in the target language starting with an input prefix v.

Prefix probabilities and right prefix probabilities for PSCFGs can be exploited to compute probability distributions for the next word or part-of-speech in left-to-right incremental translation, essentially in the same way as described by Jelinek and Lafferty (1991) for probabilistic context-free grammars, as discussed later in this paper.

Our solution to the problem of computing prefix probabilities is formulated in quite different terms from the solutions by Jelinek and Lafferty (1991) and by Stolcke (1995) for probabilistic context-free grammars. In this paper we reduce the computation of prefix probabilities for PSCFGs to the computation of inside probabilities under the same model. Computation of inside probabilities for PSCFGs is a well-known problem that can be solved using off-the-shelf algorithms that extend basic parsing algorithms. Our reduction is a novel grammar transformation, and the proof of correctness proceeds by fairly conventional techniques from formal language theory, relying on the correctness of standard methods for the computation of inside probabilities for PSCFG. This contrasts with the techniques proposed by Jelinek and Lafferty (1991) and by Stolcke (1995), which are extensions of parsing algorithms for probabilistic context-free grammars, and require considerably more involved proofs of correctness.

Our method for computing the prefix probabilities for PSCFGs runs in exponential time, since that is the running time of existing methods for computing the inside probabilities for PSCFGs. It is unlikely this can be improved, because the recognition problem for PSCFG is NP-complete, as established by Satta and Peserico (2005), and there is a straightforward reduction from the recognition problem for PSCFGs to the problem of computing the prefix probabilities for PSCFGs.

2 Definitions

In this section we introduce basic definitions related to synchronous context-free grammars and their probabilistic extension; our notation follows Satta and Peserico (2005).

Let N and Σ be sets of nonterminal and terminal symbols, respectively. In what follows we need to represent bijections between the occurrences of nonterminals in two strings over $N \cup \Sigma$. This is realized by annotating nonterminals with **indices** from an infinite set. We define $\mathcal{I}(N) = \{A^{\boxed{t}} \mid A \in N,\ t \in \mathbb{N}\}$ and $V_I = \mathcal{I}(N) \cup \Sigma$. For a string $\gamma \in V_I^*$, we write $\mathsf{index}(\gamma)$ to denote the set of all indices that appear in symbols in γ.

Two strings $\gamma_1, \gamma_2 \in V_I^*$ are **synchronous** if each index from $\mathbb{N}$ occurs at most once in γ_1 and at most once in γ_2, and $\mathsf{index}(\gamma_1) = \mathsf{index}(\gamma_2)$. Therefore γ_1, γ_2 have the general form:

$$\gamma_1 = u_{10} A_{11}^{\boxed{t_1}} u_{11} A_{12}^{\boxed{t_2}} u_{12} \cdots u_{1r-1} A_{1r}^{\boxed{t_r}} u_{1r}$$
$$\gamma_2 = u_{20} A_{21}^{\boxed{t_{\pi(1)}}} u_{21} A_{22}^{\boxed{t_{\pi(2)}}} u_{22} \cdots u_{2r-1} A_{2r}^{\boxed{t_{\pi(r)}}} u_{2r}$$

where $r \geq 0$, $u_{1i}, u_{2i} \in \Sigma^*$, $A_{1i}^{\boxed{t_i}}, A_{2i}^{\boxed{t_{\pi(i)}}} \in \mathcal{I}(N)$, $t_i \neq t_j$ for $i \neq j$, and π is a permutation of the set $\{1, \ldots, r\}$.

A **synchronous context-free grammar** (SCFG) is a tuple $G = (N, \Sigma, P, S)$, where N and Σ are finite, disjoint sets of nonterminal and terminal symbols, respectively, $S \in N$ is the start symbol and P is a finite set of **synchronous rules**. Each synchronous rule has the form $s : [A_1 \to \alpha_1,\ \ A_2 \to \alpha_2]$, where $A_1, A_2 \in N$ and where $\alpha_1, \alpha_2 \in V_I^*$ are synchronous strings. The symbol s is the label of the rule, and each rule is uniquely identified by its label. For technical reasons, we allow the existence of multiple rules that are identical apart from their labels. We refer to $A_1 \to \alpha_1$ and $A_2 \to \alpha_2$, respectively, as the left and right **components** of rule s.

Example 1 The following synchronous rules implicitly define a SCFG:

$$\begin{array}{ll} s_1: & [S \to A^{\boxed{1}}\ B^{\boxed{2}},\ S \to B^{\boxed{2}}\ A^{\boxed{1}}] \\ s_2: & [A \to aA^{\boxed{1}}b,\ A \to bA^{\boxed{1}}a] \\ s_3: & [A \to ab,\ A \to ba] \\ s_4: & [B \to cB^{\boxed{1}}d,\ B \to dB^{\boxed{1}}c] \\ s_5: & [B \to cd,\ B \to dc] \end{array}$$
□

In each step of the derivation process of a SCFG G, two nonterminals with the same index in a pair of synchronous strings are rewritten by a synchronous rule. This is done in such a way that the result is once more a pair of synchronous strings. An auxiliary notion is that of **reindexing**, which is an injective function f from $\mathbb{N}$ to $\mathbb{N}$. We extend f to V_I by letting $f(A^{\boxed{t}}) = A^{\boxed{f(t)}}$ for $A^{\boxed{t}} \in \mathcal{I}(N)$ and $f(a) = a$ for $a \in \Sigma$. We also extend f to strings in V_I^* by letting $f(\varepsilon) = \varepsilon$ and $f(X\gamma) = f(X)f(\gamma)$, for each $X \in V_I$ and $\gamma \in V_I^*$.

Let γ_1, γ_2 be synchronous strings in V_I^*. The **derive** relation $[\gamma_1,\ \gamma_2] \Rightarrow_G [\delta_1,\ \delta_2]$ holds whenever there exist an index t in $\mathsf{index}(\gamma_1) = \mathsf{index}(\gamma_2)$, a synchronous rule $s : [A_1 \to \alpha_1,\ A_2 \to \alpha_2]$ in P and some reindexing f such that:

(i) $\mathsf{index}(f(\alpha_1)) \cap (\mathsf{index}(\gamma_1) \setminus \{t\}) = \emptyset$;

(ii) $\gamma_1 = \gamma_1' A_1^{\boxed{t}} \gamma_1''$, $\gamma_2 = \gamma_2' A_2^{\boxed{t}} \gamma_2''$; and

(iii) $\delta_1 = \gamma_1' f(\alpha_1) \gamma_1''$, $\delta_2 = \gamma_2' f(\alpha_2) \gamma_2''$.

We also write $[\gamma_1,\ \gamma_2] \Rightarrow_G^s [\delta_1,\ \delta_2]$ to explicitly indicate that the derive relation holds through rule s.

Note that δ_1, δ_2 above are guaranteed to be synchronous strings, because α_1 and α_2 are synchronous strings and because of (i) above. Note also that, for a given pair $[\gamma_1,\ \gamma_2]$ of synchronous strings, an index t and a rule s, there may be infinitely many choices of reindexing f such that the above constraints are satisfied. In this paper we will not further specify the choice of f.

We say the pair $[A_1, A_2]$ of nonterminals is **linked** (in G) if there is a rule of the form $s : [A_1 \to \alpha_1,\ A_2 \to \alpha_2]$. The set of linked nonterminal pairs is denoted by $N^{[2]}$.

A derivation is a sequence $\sigma = s_1 s_2 \cdots s_d$ of synchronous rules $s_i \in P$ with $d \geq 0$ ($\sigma = \varepsilon$ for $d = 0$) such that $[\gamma_{1i-1},\ \gamma_{2i-1}] \Rightarrow_G^{s_i} [\gamma_{1i},\ \gamma_{2i}]$ for every i with $1 \leq i \leq d$ and synchronous strings $[\gamma_{1i},\ \gamma_{2i}]$ with $0 \leq i \leq d$. Throughout this paper, we always implicitly assume some canonical form for derivations in G, by demanding for instance that each step rewrites a pair of nonterminal occurrences of which the first is leftmost in the left component. When we want to focus on the specific synchronous strings being derived, we also write derivations in the form $[\gamma_{10},\ \gamma_{20}] \Rightarrow_G^\sigma [\gamma_{1d},\ \gamma_{2d}]$, and we write $[\gamma_{10},\ \gamma_{20}] \Rightarrow_G^* [\gamma_{1d},\ \gamma_{2d}]$ when σ is not further specified. The **translation** generated by a SCFG G is defined as:

$$\begin{aligned} T(G) \ = \ & \{[w_1,\ w_2] \mid [S^{\boxed{1}},\ S^{\boxed{1}}] \Rightarrow_G^* [w_1,\ w_2], \\ & w_1, w_2 \in \Sigma^*\} \end{aligned}$$

For $w_1, w_2 \in \Sigma^*$, we write $D(G, [w_1, w_2])$ to denote the set of all (canonical) derivations σ such that $[S^{\boxed{1}},\ S^{\boxed{1}}] \Rightarrow_G^\sigma [w_1,\ w_2]$.

Analogously to standard terminology for context-free grammars, we call a SCFG **reduced** if every rule occurs in at least one derivation $\sigma \in D(G, [w_1, w_2])$, for some $w_1, w_2 \in \Sigma^*$. We assume without loss of generality that the start symbol S does not occur in the right-hand side of either component of any rule.

Example 2 Consider the SCFG G from example 1. The following is a canonical derivation in G, since it is always the leftmost nonterminal occurrence in the left component that is involved in a derivation step:

$$\begin{aligned} [S^{\boxed{1}},\ S^{\boxed{1}}] & \Rightarrow_G [A^{\boxed{1}} B^{\boxed{2}}, B^{\boxed{2}} A^{\boxed{1}}] \\ & \Rightarrow_G [aA^{\boxed{3}} bB^{\boxed{2}}, B^{\boxed{2}} bA^{\boxed{3}} a] \\ & \Rightarrow_G [aaA^{\boxed{4}} bbB^{\boxed{2}}, B^{\boxed{2}} bbA^{\boxed{4}} aa] \\ & \Rightarrow_G [aaabbbB^{\boxed{2}}, B^{\boxed{2}} bbbaaa] \\ & \Rightarrow_G [aaabbbcB^{\boxed{5}} d, dB^{\boxed{5}} cbbbaaa] \\ & \Rightarrow_G [aaabbbccdd, ddccbbbaaa] \end{aligned}$$

It is not difficult to see that the generated translation is $T(G) = \{[a^p b^p c^q d^q,\ d^q c^q b^p a^p] \mid p, q \geq 1\}$. □

The size of a synchronous rule $s : [A_1 \to \alpha_1,\ A_2 \to \alpha_2]$, is defined as $|s| = |A_1 \alpha_1 A_2 \alpha_2|$. The size of G is defined as $|G| = \sum_{s \in P} |s|$.

A **probabilistic** SCFG (PSCFG) is a pair $\mathcal{G} = (G, p_G)$ where $G = (N, \Sigma, P, S)$ is a SCFG and p_G is a function from P to real numbers in $[0, 1]$. We say that $\mathcal{G}$ is **proper** if for each pair $[A_1, A_2] \in N^{[2]}$ we have:

$$\sum_{s:[A_1 \to \alpha_1,\ A_2 \to \alpha_2]} p_G(s) \ = \ 1$$

Intuitively, properness ensures that where a pair of nonterminals in two synchronous strings can be rewritten, there is a probability distribution over the applicable rules.

For a (canonical) derivation $\sigma = s_1 s_2 \cdots s_d$, we define $p_G(\sigma) = \prod_{i=1}^d p_G(s_i)$. For $w_1, w_2 \in \Sigma^*$, we also define:

$$p_G([w_1, w_2]) \ = \sum_{\sigma \in D(G, [w_1, w_2])} p_G(\sigma) \qquad (1)$$

We say a PSCFG is **consistent** if p_G defines a probability distribution over the translation, or formally:

$$\sum_{w_1, w_2} p_G([w_1, w_2]) = 1$$

If the grammar is reduced, proper and consistent, then also:

$$\sum_{\substack{w_1,w_2 \in \Sigma^*,\ \sigma \in P^* \\ \text{s.t. } [A_1^{\boxed{1}},\, A_2^{\boxed{1}}] \Rightarrow_G^{\sigma} [w_1,\, w_2]}} p_G(\sigma) \;=\; 1$$

for every pair $[A_1, A_2] \in N^{[2]}$. The proof is identical to that of the corresponding fact for probabilistic context-free grammars.

3 Effective PSCFG parsing

If $w = a_1 \cdots a_n$ then the expression $w[i, j]$, with $0 \leq i \leq j \leq n$, denotes the substring $a_{i+1} \cdots a_j$ (if $i = j$ then $w[i, j] = \varepsilon$). In this section, we assume the input is the pair $[w_1, w_2]$ of terminal strings. The task of a **recognizer** for SCFG G is to decide whether $[w_1, w_2] \in T(G)$.

We present a general algorithm for solving the above problem in terms of the specification of a deduction system, following Shieber et al. (1995). The items that are constructed by the system have the form $[m_1, A_1, m_1';\ m_2, A_2, m_2']$, where $[A_1, A_2] \in N^{[2]}$ and where m_1, m_1', m_2, m_2' are non-negative integers such that $0 \leq m_1 \leq m_1' \leq |w_1|$ and $0 \leq m_2 \leq m_2' \leq |w_2|$. Such an item can be derived by the deduction system if and only if:

$$[A_1^{\boxed{1}},\, A_2^{\boxed{1}}] \;\; \Rightarrow_G^* \;\; [w_1[m_1, m_1'],\ w_2[m_2, m_2']]$$

The deduction system has one inference rule, shown in figure 1. One of its side conditions has a synchronous rule in P of the form:

$$\begin{aligned} s : [&A_1 \rightarrow u_{10} A_{11}^{\boxed{t_1}} u_{11} \cdots u_{1r-1} A_{1r}^{\boxed{t_r}} u_{1r}, \\ &A_2 \rightarrow u_{20} A_{21}^{\boxed{t_{\pi(1)}}} u_{21} \cdots u_{2r-1} A_{2r}^{\boxed{t_{\pi(r)}}} u_{2r}] \quad (2) \end{aligned}$$

Observe that, in the right-hand side of the two rule components above, nonterminals A_{1i} and $A_{2\pi^{-1}(i)}$, $1 \leq i \leq r$, have both the same index. More precisely, A_{1i} has index t_i and $A_{2\pi^{-1}(i)}$ has index $t_{i'}$ with $i' = \pi(\pi^{-1}(i)) = i$. Thus the nonterminals in each antecedent item in figure 1 form a linked pair.

We now turn to a computational analysis of the above algorithm. In the inference rule in figure 1 there are $2(r + 1)$ variables that can be bound to positions in w_1, and as many that can be bound to positions in w_2. However, the side conditions imply $m_{ij}' = m_{ij} + |u_{ij}|$, for $i \in \{1, 2\}$ and $0 \leq j \leq r$, and therefore the number of free variables is only $r + 1$ for each component. By standard complexity analysis of deduction systems, for example following McAllester (2002), the time complexity of a straightforward implementation of the recognition algorithm is $\mathcal{O}(|P| \cdot |w_1|^{r_{\max}+1} \cdot |w_2|^{r_{\max}+1})$, where $r_{\max}$ is the maximum number of right-hand side nonterminals in either component of a synchronous rule. The algorithm therefore runs in exponential time, when the grammar G is considered as part of the input. Such computational behavior seems unavoidable, since the recognition problem for SCFG is NP-complete, as reported by Satta and Peserico (2005). See also Gildea and Stefankovic (2007) and Hopkins and Langmead (2010) for further analysis of the upper bound above.

The recognition algorithm above can easily be turned into a parsing algorithm by letting an implementation keep track of which items were derived from which other items, as instantiations of the consequent and the antecedents, respectively, of the inference rule in figure 1.

A probabilistic parsing algorithm that computes $p_G([w_1, w_2])$, defined in (1), can also be obtained from the recognition algorithm above, by associating each item with a probability. To explain the basic idea, let us first assume that each item can be inferred in finitely many ways by the inference rule in figure 1. Each instantiation of the inference rule should be associated with a term that is computed by multiplying the probability of the involved rule s and the product of all probabilities previously associated with the instantiations of the antecedents. The probability associated with an item is then computed as the sum of each term resulting from some instantiation of an inference rule deriving that item. This is a generalization to PSCFG of the inside algorithm defined for probabilistic context-free grammars (Manning and Schütze, 1999), and we can show that the probability associated with item $[0, S, |w_1|\,;\ 0, S, |w_2|]$ provides the desired value $p_G([w_1, w_2])$. We refer to the procedure sketched above as the **inside** algorithm for PSCFGs.

However, this simple procedure fails if there are cyclic dependencies, whereby the derivation of an item involves a proper subderivation of the same item. Cyclic dependencies can be excluded if it can

$$\frac{\begin{matrix}[m'_{10}, A_{11}, m_{11};\ m'_{2\pi^{-1}(1)-1}, A_{2\pi^{-1}(1)}, m_{2\pi^{-1}(1)}] \\ \vdots \\ [m'_{1r-1}, A_{1r}, m_{1r};\ m'_{2\pi^{-1}(r)-1}, A_{2\pi^{-1}(r)}, m_{2\pi^{-1}(r)}]\end{matrix}}{[m_{10}, A_1, m'_{1r};\ m_{20}, A_2, m'_{2r}]} \quad \begin{cases} s{:}[A_1 \to u_{10}A_{11}^{\boxed{t_1}}u_{11}\cdots u_{1r-1}A_{1r}^{\boxed{t_r}}u_{1r}, \\ \quad A_2 \to u_{20}A_{21}^{\boxed{t_{\pi(1)}}}u_{21}\cdots u_{2r-1}A_{2r}^{\boxed{t_{\pi(r)}}}u_{2r}] \in P, \\ w_1[m_{10}, m'_{10}] = u_{10}, \quad w_2[m_{20}, m'_{20}] = u_{20}, \\ \qquad \vdots \qquad\qquad\qquad \vdots \\ w_1[m_{1r}, m'_{1r}] = u_{1r}, \quad w_2[m_{2r}, m'_{2r}] = u_{2r} \end{cases}$$

Figure 1: SCFG recognition, by a deduction system consisting of a single inference rule.

be guaranteed that, in figure 1, $m'_{1r} - m_{10}$ is greater than $m_{1j} - m'_{1j-1}$ for each j $(1 \leq j \leq r)$, or $m'_{2r} - m_{20}$ is greater than $m_{2j} - m'_{2j-1}$ for each j $(1 \leq j \leq r)$.

Consider again a synchronous rule s of the form in (2). We say s is an **epsilon** rule if $r = 0$ and $u_{10} = u_{20} = \epsilon$. We say s is a **unit** rule if $r = 1$ and $u_{10} = u_{11} = u_{20} = u_{21} = \epsilon$. Similarly to context-free grammars, absence of epsilon rules and unit rules guarantees that there are no cyclic dependencies between items and in this case the inside algorithm correctly computes $p_G([w_1, w_2])$.

Epsilon rules can be eliminated from PSCFGs by a grammar transformation that is very similar to the transformation eliminating epsilon rules from a probabilistic context-free grammar (Abney et al., 1999). This is sketched in what follows. We first compute the set of all **nullable** linked pairs of nonterminals of the underlying SCFG, that is, the set of all $[A_1, A_2] \in N^{[2]}$ such that $[A_1^{\boxed{1}},\ A_2^{\boxed{1}}] \Rightarrow_G^* [\varepsilon,\ \varepsilon]$. This can be done in linear time $\mathcal{O}(|G|)$ using essentially the same algorithm that identifies nullable nonterminals in a context-free grammar, as presented for instance by Sippu and Soisalon-Soininen (1988).

Next, we identify all occurrences of nullable pairs $[A_1, A_2]$ in the right-hand side components of a rule s, such that A_1 and A_2 have the same index. For every possible choice of a subset $\mathcal{U}$ of these occurrences, we add to our grammar a new rule $s_{\mathcal{U}}$ constructed by omitting all of the nullable occurrences in $\mathcal{U}$. The probability of $s_{\mathcal{U}}$ is computed as the probability of s multiplied by terms of the form:

$$\sum_{\sigma \text{ s.t. } [A_1^{\boxed{1}}, A_2^{\boxed{1}}] \Rightarrow_G^{\sigma} [\varepsilon,\ \varepsilon]} p_G(\sigma) \qquad (3)$$

for every pair $[A_1, A_2]$ in $\mathcal{U}$. After adding these extra rules, which in effect circumvents the use of epsilon-generating subderivations, we can safely remove all epsilon rules, with the only exception of a possible rule of the form $[S \to \epsilon, S \to \epsilon]$. The translation and the associated probability distribution in the resulting grammar will be the same as those in the source grammar.

One problem with the above construction is that we have to create new synchronous rules $s_{\mathcal{U}}$ for each possible choice of subset $\mathcal{U}$. In the worst case, this may result in an exponential blow-up of the source grammar. In the case of context-free grammars, this is usually circumvented by casting the rules in binary form prior to epsilon rule elimination. However, this is not possible in our case, since SCFGs do not allow normal forms with a constant bound on the length of the right-hand side of each component. This follows from a result due to Aho and Ullman (1969) for a formalism called syntax directed translation schemata, which is a syntactic variant of SCFGs.

An additional complication with our construction is that finding any of the values in (3) may involve solving a system of non-linear equations, similarly to the case of probabilistic context-free grammars; see again Abney et al. (1999), and Stolcke (1995). Approximate solution of such systems might take exponential time, as pointed out by Kiefer et al. (2007).

Notwithstanding the worst cases mentioned above, there is a special case that can be easily dealt with. Assume that, for each nullable pair $[A_1, A_2]$ in G we have that $[A_1^{\boxed{1}},\ A_2^{\boxed{1}}] \Rightarrow_G^* [w_1,\ w_2]$ does not hold for any w_1 and w_2 with $w_1 \neq \varepsilon$ or $w_2 \neq \varepsilon$. Then each of the values in (3) is guaranteed to be 1, and furthermore we can remove the instances of the nullable pairs in the source rule s all at the same time. This means that the overall construction of elimination of nullable rules from G can be implemented in linear time $|G|$. It is this special case that we will encounter in section 4.

After elimination of epsilon rules, one can eliminate unit rules. We define $C^{\text{unit}}([A_1, A_2], [B_1, B_2])$ as the sum of the probabilities of all derivations deriving $[B_1, B_2]$ from $[A_1, A_2]$ with arbitrary indices, or more precisely:

$$\sum_{\substack{\sigma \in P^* \text{ s.t. } \exists t \in \mathbb{N}, \\ [A_1^{\boxed{1}}, A_2^{\boxed{1}}] \Rightarrow_G^{\sigma} [B_1^{\boxed{t}}, B_2^{\boxed{t}}]}} p_G(\sigma)$$

Note that $[A_1, A_2]$ may be equal to $[B_1, B_2]$ and σ may be ε, in which case $C^{\text{unit}}([A_1, A_2], [B_1, B_2])$ is at least 1, but it may be larger if there are unit rules. Therefore $C^{\text{unit}}([A_1, A_2], [B_1, B_2])$ should *not* be seen as a probability.

Consider a pair $[A_1, A_2] \in N^{[2]}$ and let all unit rules with left-hand sides A_1 and A_2 be:

$$\begin{aligned} s_1 &: [A_1, A_2] \rightarrow [A_{11}^{\boxed{t_1}}, A_{21}^{\boxed{t_1}}] \\ &\vdots \\ s_m &: [A_1, A_2] \rightarrow [A_{1m}^{\boxed{t_m}}, A_{2m}^{\boxed{t_m}}] \end{aligned}$$

The values of $C^{\text{unit}}(\cdot, \cdot)$ are related by the following:

$$\begin{aligned} & C^{\text{unit}}([A_1, A_2], [B_1, B_2]) = \\ & \qquad \delta([A_1, A_2] = [B_1, B_2]) + \\ & \qquad \sum_i p_G(s_i) \cdot C^{\text{unit}}([A_{1i}, A_{2i}], [B_1, B_2]) \end{aligned}$$

where $\delta([A_1, A_2] = [B_1, B_2])$ is defined to be 1 if $[A_1, A_2] = [B_1, B_2]$ and 0 otherwise. This forms a system of linear equations in the unknown variables $C^{\text{unit}}(\cdot, \cdot)$. Such a system can be solved in polynomial time in the number of variables, for example using Gaussian elimination.

The elimination of unit rules starts with adding a rule $s' : [A_1 \rightarrow \alpha_1, A_2 \rightarrow \alpha_2]$ for each non-unit rule $s : [B_1 \rightarrow \alpha_1, B_2 \rightarrow \alpha_2]$ and pair $[A_1, A_2]$ such that $C^{\text{unit}}([A_1, A_2], [B_1, B_2]) > 0$. We assign to the new rule s' the probability $p_G(s) \cdot C^{\text{unit}}([A_1, A_2], [B_1, B_2])$. The unit rules can now be removed from the grammar. Again, in the resulting grammar the translation and the associated probability distribution will be the same as those in the source grammar. The new grammar has size $\mathcal{O}(|G|^2)$, where G is the input grammar. The time complexity is dominated by the computation of the solution of the linear system of equations. This computation takes cubic time in the number of variables. The number of variables in this case is $\mathcal{O}(|G|^2)$, which makes the running time $\mathcal{O}(|G|^6)$.

4 Prefix probabilities

The **joint prefix probability** $p_G^{\text{prefix}}([v_1, v_2])$ of a pair $[v_1, v_2]$ of terminal strings is the sum of the probabilities of all pairs of strings that have v_1 and v_2, respectively, as their prefixes. Formally:

$$p_G^{\text{prefix}}([v_1, v_2]) \;=\; \sum_{w_1, w_2 \in \Sigma^*} p_G([v_1 w_1, v_2 w_2])$$

At first sight, it is not clear this quantity can be effectively computed, as it involves a sum over infinitely many choices of w_1 and w_2. However, analogously to the case of context-free prefix probabilities (Jelinek and Lafferty, 1991), we can isolate two parts in the computation. One part involves infinite sums, which are independent of the input strings v_1 and v_2, and can be precomputed by solving a system of linear equations. The second part does rely on v_1 and v_2, and involves the actual evaluation of $p_G^{\text{prefix}}([v_1, v_2])$. This second part can be realized effectively, on the basis of the precomputed values from the first part.

In order to keep the presentation simple, and to allow for simple proofs of correctness, we solve the problem in a modular fashion. First, we present a transformation from a PSCFG $\mathcal{G} = (G, p_G)$, with $G = (N, \Sigma, P, S)$, to a PSCFG $\mathcal{G}_{\text{prefix}} = (G_{\text{prefix}}, p_{G_{\text{prefix}}})$, with $G_{\text{prefix}} = (N_{\text{prefix}}, \Sigma, P_{\text{prefix}}, S^{\downarrow})$. The latter grammar derives all possible pairs $[v_1, v_2]$ such that $[v_1 w_1, v_2 w_2]$ can be derived from G, for some w_1 and w_2. Moreover, $p_{G_{\text{prefix}}}([v_1, v_2]) = p_G^{\text{prefix}}([v_1, v_2])$, as will be verified later.

Computing $p_{G_{\text{prefix}}}([v_1, v_2])$ directly using a generic probabilistic parsing algorithm for PSCFGs is difficult, due to the presence of epsilon rules and unit rules. The next step will be to transform $\mathcal{G}_{\text{prefix}}$ into a third grammar $\mathcal{G}'_{\text{prefix}}$ by eliminating epsilon rules and unit rules from the underlying SCFG, and preserving the probability distribution over pairs of strings. Using $\mathcal{G}'_{\text{prefix}}$ one can then effectively apply generic probabilistic parsing algorithms for PSCFGs, such as the inside algorithm discussed in section 3, in order to compute the desired prefix probabilities for the source PSCFG $\mathcal{G}$.

For each nonterminal A in the source SCFG G, the grammar G_{prefix} contains three nonterminals, namely A itself, $A^{\downarrow}$ and A^{ε}. The meaning of A remains unchanged, whereas $A^{\downarrow}$ is intended to generate a string that is a suffix of a known prefix v_1 or v_2. Nonterminals A^{ε} generate only the empty string, and are used to simulate the generation by G of infixes of the unknown suffix w_1 or w_2. The two left-hand sides of a synchronous rule in G_{prefix} can contain different combinations of nonterminals of the forms A, $A^{\downarrow}$, or A^{ε}. The start symbol of G_{prefix} is $S^{\downarrow}$. The structure of the rules from the source grammar is largely retained, except that some terminal symbols are omitted in order to obtain the intended interpretation of $A^{\downarrow}$ and A^{ε}.

In more detail, let us consider a synchronous rule $s : [A_1 \to \alpha_1,\ \ A_2 \to \alpha_2]$ from the source grammar, where for $i \in \{1, 2\}$ we have:

$$\alpha_i = u_{i0} A_{i1}^{\boxed{t_{i1}}} u_{i1} \cdots u_{ir-1} A_{ir}^{\boxed{t_{ir}}} u_{ir}$$

The transformed grammar then contains a large number of rules, each of which is of the form $s' : [B_1 \to \beta_1,\ \ B_2 \to \beta_2]$, where $B_i \to \beta_i$ is of one of three forms, namely $A_i \to \alpha_i$, $A_i^{\downarrow} \to \alpha_i^{\downarrow}$ or $A_i^{\varepsilon} \to \alpha_i^{\varepsilon}$, where $\alpha_i^{\downarrow}$ and α_i^{ε} are explained below. The choices for $i = 1$ and for $i = 2$ are independent, so that we can have $3 * 3 = 9$ kinds of synchronous rules, to be further subdivided in what follows. A unique label s' is produced for each new rule, and the probability of each new rule equals that of s.

The right-hand side α_i^{ε} is constructed by omitting all terminals and propagating downwards the ε superscript, resulting in:

$$\alpha_i^{\varepsilon} = A_{i1}^{\varepsilon \boxed{t_{i1}}} \cdots A_{ir}^{\varepsilon \boxed{t_{ir}}}$$

It is more difficult to define $\alpha_i^{\downarrow}$. In fact, there can be a number of choices for $\alpha_i^{\downarrow}$ and, for each choice, the transformed grammar contains an instance of the synchronous rule $s' : [B_1 \to \beta_1,\ \ B_2 \to \beta_2]$ as defined above. The reason why different choices need to be considered is because the boundary between the known prefix v_i and the unknown suffix w_i can occur at different positions, either within a terminal string u_{ij} or else further down in a subderivation involving A_{ij}. In the first case, we have for some j $(0 \leq j \leq r)$:

$$\begin{aligned}\alpha_i^{\downarrow} = {} & u_{i0} A_{i1}^{\boxed{t_{i1}}} u_{i1} A_{i2}^{\boxed{t_{i2}}} \cdots \\ & u_{ij-1} A_{ij}^{\boxed{t_{ij}}} u'_{ij} A_{ij+1}^{\varepsilon \boxed{t_{ij+1}}} A_{ij+2}^{\varepsilon \boxed{t_{ij+2}}} \cdots A_{ir}^{\varepsilon \boxed{t_{ir}}}\end{aligned}$$

where u'_{ij} is a choice of a prefix of u_{ij}. In words, the known prefix ends after u'_{ij} and, thereafter, no more terminals are generated. We demand that u'_{ij} must not be the empty string, unless $A_i = S$ and $j = 0$. The reason for this restriction is that we want to avoid an overlap with the second case. In this second case, we have for some j $(1 \leq j \leq r)$:

$$\begin{aligned}\alpha_i^{\downarrow} = {} & u_{i0} A_{i1}^{\boxed{t_{i1}}} u_{i1} A_{i2}^{\boxed{t_{i2}}} \cdots \\ & u_{ij-1} A_{ij}^{\downarrow \boxed{t_{ij}}} A_{ij+1}^{\varepsilon \boxed{t_{ij+1}}} A_{ij+2}^{\varepsilon \boxed{t_{ij+2}}} \cdots A_{ir}^{\varepsilon \boxed{t_{ir}}}\end{aligned}$$

Here the known prefix of the input ends within a subderivation involving A_{ij}, and further to the right no more terminals are generated.

Example 3 Consider the synchronous rule $s : [A \to aB^{\boxed{1}} bc\, C^{\boxed{2}} d, D \to ef E^{\boxed{2}} F^{\boxed{1}}]$. The first component of a synchronous rule derived from this can be one of the following eight:

$$\begin{aligned}
& A^{\varepsilon} \to B^{\varepsilon \boxed{1}} C^{\varepsilon \boxed{2}} \\
& A^{\downarrow} \to aB^{\varepsilon \boxed{1}} C^{\varepsilon \boxed{2}} \\
& A^{\downarrow} \to aB^{\downarrow \boxed{1}} C^{\varepsilon \boxed{2}} \\
& A^{\downarrow} \to aB^{\boxed{1}}\, b\, C^{\varepsilon \boxed{2}} \\
& A^{\downarrow} \to aB^{\boxed{1}}\, bc\, C^{\varepsilon \boxed{2}} \\
& A^{\downarrow} \to aB^{\boxed{1}}\, bc\, C^{\downarrow \boxed{2}} \\
& A^{\downarrow} \to aB^{\boxed{1}}\, bc\, C^{\boxed{2}} d \\
& A \to aB^{\boxed{1}}\, bc\, C^{\boxed{2}} d
\end{aligned}$$

The second component can be one of the following six:

$$\begin{aligned}
& D^{\varepsilon} \to E^{\varepsilon \boxed{2}} F^{\varepsilon \boxed{1}} \\
& D^{\downarrow} \to eE^{\varepsilon \boxed{2}} F^{\varepsilon \boxed{1}} \\
& D^{\downarrow} \to ef E^{\varepsilon \boxed{2}} F^{\varepsilon \boxed{1}} \\
& D^{\downarrow} \to ef E^{\downarrow \boxed{2}} F^{\varepsilon \boxed{1}} \\
& D^{\downarrow} \to ef E^{\boxed{2}} F^{\downarrow \boxed{1}} \\
& D \to ef E^{\boxed{2}} F^{\boxed{1}}
\end{aligned}$$

In total, the transformed grammar will contain $8 * 6 = 48$ synchronous rules derived from s. $\square$

For each synchronous rule s, the above grammar transformation produces $\mathcal{O}(|s|)$ left rule components and as many right rule components. This means the number of new synchronous rules is $\mathcal{O}(|s|^2)$, and the size of each such rule is $\mathcal{O}(|s|)$. If we sum $\mathcal{O}(|s|^3)$ for every rule s we obtain a time and space complexity of $\mathcal{O}(|G|^3)$.

We now investigate formal properties of our grammar transformation, in order to relate it to prefix probabilities. We define the relation $\vdash$ between P and P_{prefix} such that $s \vdash s'$ if and only if s' was obtained from s by the transformation described above. This is extended in a natural way to derivations, such that $s_1 \cdots s_d \vdash s'_1 \cdots s'_{d'}$ if and only if $d = d'$ and $s_i \vdash s'_i$ for each i ($1 \leq i \leq d$).

The formal relation between G and G_{prefix} is revealed by the following two lemmas.

Lemma 1 *For each v_1, v_2, w_1, $w_2 \in \Sigma^*$ and $\sigma \in P^*$ such that $[S, S] \Rightarrow^{\sigma}_{G} [v_1w_1, v_2w_2]$, there is a unique $\sigma' \in P^*_{\text{prefix}}$ such that $[S^{\downarrow}, S^{\downarrow}] \Rightarrow^{\sigma'}_{G_{\text{prefix}}} [v_1, v_2]$ and $\sigma \vdash \sigma'$.* $\square$

Lemma 2 *For each v_1, $v_2 \in \Sigma^*$ and derivation $\sigma' \in P^*_{\text{prefix}}$ such that $[S^{\downarrow}, S^{\downarrow}] \Rightarrow^{\sigma'}_{G_{\text{prefix}}} [v_1, v_2]$, there is a unique $\sigma \in P^*$ and unique w_1, $w_2 \in \Sigma^*$ such that $[S, S] \Rightarrow^{\sigma}_{G} [v_1w_1, v_2w_2]$ and $\sigma \vdash \sigma'$.* $\square$

The only non-trivial issue in the proof of Lemma 1 is the uniqueness of σ'. This follows from the observation that the length of v_1 in v_1w_1 uniquely determines how occurrences of left components of rules in P found in σ are mapped to occurrences of left components of rules in P_{prefix} found in σ'. The same applies to the length of v_2 in v_2w_2 and the right components.

Lemma 2 is easy to prove as the structure of the transformation ensures that the terminals that are in rules from P but not in the corresponding rules from P_{prefix} occur at the end of a string v_1 (and v_2) to form the longer string v_1w_1 (and v_2w_2, respectively).

The transformation also ensures that $s \vdash s'$ implies $p_G(s) = p_{G_{\text{prefix}}}(s')$. Therefore $\sigma \vdash \sigma'$ implies $p_G(\sigma) = p_{G_{\text{prefix}}}(\sigma')$. By this and Lemmas 1 and 2 we may conclude:

Theorem 1 $p_{G_{\text{prefix}}}([v_1, v_2]) = p^{\text{prefix}}_{G}([v_1, v_2])$. $\square$

Because of the introduction of rules with left-hand sides of the form A^{ε} in both the left and right components of synchronous rules, it is not straightforward to do effective probabilistic parsing with the grammar $\mathcal{G}_{\text{prefix}}$. We can however apply the transformations from section 3 to eliminate epsilon rules and thereafter eliminate unit rules, in a way that leaves the derived string pairs and their probabilities unchanged.

The simplest case is when the source grammar $\mathcal{G}$ is reduced, proper and consistent, and has no epsilon rules. The only nullable pairs of nonterminals in $\mathcal{G}_{\text{prefix}}$ will then be of the form $[A^{\varepsilon}_1, A^{\varepsilon}_2]$. Consider such a pair $[A^{\varepsilon}_1, A^{\varepsilon}_2]$. Because of reduction, properness and consistency of $\mathcal{G}$ we have:

$$\sum_{\substack{w_1,w_2 \in \Sigma^*,\ \sigma \in P^* \text{ s.t.} \\ [A_1^{\boxed{1}},\, A_2^{\boxed{1}}] \Rightarrow^{\sigma}_{G} [w_1,\, w_2]}} p_G(\sigma) \quad = \quad 1$$

Because of the structure of the grammar transformation by which $\mathcal{G}_{\text{prefix}}$ was obtained from $\mathcal{G}$, we also have:

$$\sum_{\substack{\sigma \in P^* \text{ s.t.} \\ [A_1^{\varepsilon\boxed{1}},\, A_2^{\varepsilon\boxed{1}}] \Rightarrow^{\sigma}_{G_{\text{prefix}}} [\varepsilon,\, \varepsilon]}} p_{G_{\text{prefix}}}(\sigma) = 1$$

Therefore pairs of occurrences of A^{ε}_1 and A^{ε}_2 with the same index in synchronous rules of G_{prefix} can be systematically removed without affecting the probability of the resulting rule, as outlined in section 3. Thereafter, unit rules can be removed to allow parsing by the inside algorithm for PSCFGs.

Following the computational analyses for all of the constructions presented in section 3, and for the grammar transformation discussed in this section, we can conclude that the running time of the proposed algorithm for the computation of prefix probabilities is dominated by the running time of the inside algorithm, which in the worst case is exponential in $|G|$. This result is not unexpected, as already pointed out in the introduction, since the recognition problem for PSCFGs is NP-complete, as established by Satta and Peserico (2005), and there is a straightforward reduction from the recognition problem for PSCFGs to the problem of computing the prefix probabilities for PSCFGs.

One should add that, in real world machine translation applications, it has been observed that recognition (and computation of inside probabilities) for SCFGs can typically be carried out in low-degree polynomial time, and the worst cases mentioned above are not observed with real data. Further discussion on this issue is due to Zhang et al. (2006).

5 Discussion

We have shown that the computation of joint prefix probabilities for PSCFGs can be reduced to the computation of inside probabilities for the same model. Our reduction relies on a novel grammar transformation, followed by elimination of epsilon rules and unit rules.

Next to the joint prefix probability, we can also consider the **right prefix probability**, which is defined by:

$$p_G^{\mathrm{r-prefix}}([v_1, v_2]) \quad = \quad \sum_w p_G([v_1, v_2 w])$$

In words, the entire left string is given, along with a prefix of the right string, and the task is to sum the probabilities of all string pairs for different suffixes following the given right prefix. This can be computed as a special case of the joint prefix probability. Concretely, one can extend the input and the grammar by introducing an end-of-sentence marker \$. Let G' be the underlying SCFG grammar after the extension. Then:

$$p_G^{\mathrm{r-prefix}}([v_1, v_2]) = p_{G'}^{\mathrm{prefix}}([v_1\$, v_2])$$

Prefix probabilities and right prefix probabilities for PSCFGs can be exploited to compute probability distributions for the next word or part-of-speech in left-to-right incremental translation of speech, or alternatively as a predictive tool in applications of interactive machine translation, of the kind described by Foster et al. (2002). We provide some technical details here, generalizing to PSCFGs the approach by Jelinek and Lafferty (1991).

Let $\mathcal{G} = (G, p_G)$ be a PSCFG, with Σ the alphabet of terminal symbols. We are interested in the probability that the next terminal in the target translation is $a \in \Sigma$, after having processed a prefix v_1 of the source sentence and having produced a prefix v_2 of the target translation. This can be computed as:

$$p_G^{\mathrm{r-word}}(a \mid [v_1, v_2]) \quad = \quad \frac{p_G^{\mathrm{prefix}}([v_1, v_2 a])}{p_G^{\mathrm{prefix}}([v_1, v_2])}$$

Two considerations are relevant when applying the above formula in practice. First, the computation of $p_G^{\mathrm{prefix}}([v_1, v_2 a])$ need not be computed from scratch if $p_G^{\mathrm{prefix}}([v_1, v_2])$ has been computed already. Because of the tabular nature of the inside algorithm, one can extend the table for $p_G^{\mathrm{prefix}}([v_1, v_2])$ by adding new entries to obtain the table for $p_G^{\mathrm{prefix}}([v_1, v_2 a])$. The same holds for the computation of $p_G^{\mathrm{prefix}}([v_1 b, v_2])$.

Secondly, the computation of $p_G^{\mathrm{prefix}}([v_1, v_2 a])$ for all possible $a \in \Sigma$ may be impractical. However, one may also compute the probability that the next part-of-speech in the target translation is A. This can be realised by adding a rule $s' : [B \rightarrow b, A \rightarrow c_A]$ for each rule $s : [B \rightarrow b, A \rightarrow a]$ from the source grammar, where A is a nonterminal representing a part-of-speech and c_A is a (pre-)terminal specific to A. The probability of s' is the same as that of s. If G' is the underlying SCFG after adding such rules, then the required value is $p_{G'}^{\mathrm{prefix}}([v_1, v_2\, c_A])$.

One variant of the definitions presented in this paper is the notion of *infix* probability, which is useful in island-driven speech translation. Here we are interested in the probability that any string in the source language with infix v_1 is translated into any string in the target language with infix v_2. However, just as infix probabilities are difficult to compute for probabilistic context-free grammars (Corazza et al., 1991; Nederhof and Satta, 2008) so (joint) infix probabilities are difficult to compute for PSCFGs. The problem lies in the possibility that a given infix may occur more than once in a string in the language. The computation of infix probabilities can be reduced to that of solving non-linear systems of equations, which can be approximated using for instance Newton's algorithm. However, such a system of equations is built from the input strings, which entails that the computational effort of solving the system primarily affects parse time rather than parser-generation time.

References

S. Abney, D. McAllester, and F. Pereira. 1999. Relating probabilistic grammars and automata. In *37th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 542–549, Maryland, USA, June.

A.V. Aho and J.D. Ullman. 1969. Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, 3:37–56.

Z. Chi. 1999. Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, 25(1):131–160.

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

A. Corazza, R. De Mori, R. Gretter, and G. Satta. 1991. Computation of probabilities for an island-driven parser. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):936–950.

G. Foster, P. Langlais, and G. Lapalme. 2002. User-friendly text prediction for translators. In *Conference on Empirical Methods in Natural Language Processing*, pages 148–155, University of Pennsylvania, Philadelphia, PA, USA, July.

M. Galley, M. Hopkins, K. Knight, and D. Marcu. 2004. What's in a translation rule? In *HLT-NAACL 2004, Proceedings of the Main Conference*, Boston, Massachusetts, USA, May.

D. Gildea and D. Stefankovic. 2007. Worst-case synchronous grammar rules. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics, Proceedings of the Main Conference*, pages 147–154, Rochester, New York, USA, April.

M. Hopkins and G. Langmead. 2010. SCFG decoding without binarization. In *Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 646–655, October.

F. Jelinek and J.D. Lafferty. 1991. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17(3):315–323.

S. Kiefer, M. Luttenberger, and J. Esparza. 2007. On the convergence of Newton's method for monotone systems of polynomial equations. In *Proceedings of the 39th ACM Symposium on Theory of Computing*, pages 217–266.

C.D. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

D. McAllester. 2002. On the complexity analysis of static analyses. *Journal of the ACM*, 49(4):512–537.

M.-J. Nederhof and G. Satta. 2008. Computing partition functions of PCFGs. *Research on Language and Computation*, 6(2):139–162.

G. Satta and E. Peserico. 2005. Some computational complexity results for synchronous context-free grammars. In *Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 803–810.

S.M. Shieber, Y. Schabes, and F.C.N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–36.

S. Sippu and E. Soisalon-Soininen. 1988. *Parsing Theory, Vol. I: Languages and Parsing*, volume 15 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag.

A. Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):167–201.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 256–263, New York, USA, June.