A General Technique to Train Language Models on Language Models

Mark-Jan Nederhof*
University of Groningen

We show that under certain conditions, a language model can be trained on the basis of a second language model. The main instance of the technique trains a finite automaton on the basis of a probabilistic context-free grammar, such that the Kullback-Leibler distance between grammar and trained automaton is provably minimal. This is a substantial generalization of an existing algorithm to train an n*-gram model on the basis of a probabilistic context-free grammar.*

1. Introduction

In this article, the term *language model* is used to refer to any description that assigns probabilities to strings over a certain alphabet. Language models have important applications in natural language processing, and in particular, in speech recognition systems (Manning and Schütze 1999).

Language models often consist of a symbolic description of a language, such as a finite automaton (FA) or a context-free grammar (CFG), extended by a probability assignment to, for example, the transitions of the FA or the rules of the CFG, by which we obtain a probabilistic finite automaton (PFA) or probabilistic context-free grammar (PCFG), respectively. For certain applications, one may first determine the symbolic part of the automaton or grammar and in a second phase try to find reliable probability estimates for the transitions or rules. The current article is involved with the second problem, that of extending FAs or CFGs to become PFAs or PCFGs. We refer to this process as *training*.

Training is often done on the basis of a corpus of actual language use in a certain domain. If each sentence in this corpus is annotated by a list of transitions of an FA recognizing the sentence or a parse tree for a CFG generating the sentence, then training may consist simply in *relative frequency estimation*. This means that we estimate probabilities of transitions or rules by counting their frequencies in the corpus, relative to the frequencies of the start states of transitions or to the frequencies of the left-hand side nonterminals of rules, respectively. By this estimation, the likelihood of the corpus is maximized.

The technique we introduce in this article is different in that training is done on the basis not of a finite corpus, but of an input language model. Our goal is to find estimations for the probabilities of transitions or rules of the input FA or CFG such that

* Faculty of Arts, Humanities Computing, P.O. Box 716, NL-9700 AS Groningen, The Netherlands. E-mail: markjan@let.rug.nl.

Submission received: 20th January 2004; Revised submission received: 5th August 2004; Accepted for publication: 19th September 2004

the resulting PFA or PCFG approximates the input language model as well as possible, or more specifically, such that the Kullback-Leibler (KL) distance (or relative entropy) between the input model and the trained model is minimized. The input FA or CFG to be trained may be structurally unrelated to the input language model.

This technique has several applications. One is an extension with probabilities of existing work on approximation of CFGs by means of FAs (Nederhof 2000). The motivation for this work was that application of FAs is generally less costly than application of CFGs, which is an important benefit when the input is very large, as is often the case in, for example, speech recognition systems. The practical relevance of this work was limited, however, by the fact that in practice one is more interested in the probabilities of sentences than in a purely Boolean distinction between grammatical and ungrammatical sentences.

Several approaches were discussed by Mohri and Nederhof (2001) to extend this work to approximation of PCFGs by means of PFAs. A first approach is to directly map rules with attached probabilities to transitions with attached probabilities. Although this is computationally the easiest approach, the resulting PFA may be a very inaccurate approximation of the probability distribution described by the input PCFG. In particular, there may be assignments of probabilities to the transitions of the same FA that lead to more accurate approximating language models.

A second approach is to train the approximating FA by means of a corpus. If the input PCFG was itself obtained by training on a corpus, then we already possess training material. However, this may not always be the case, and no training material may be available. Furthermore, as a determinized approximating FA may be much larger than the input PCFG, the sparse-data problem may be more severe for the automaton than it was for the grammar.[1] Hence, even if sufficient material was available to train the CFG, it may not be sufficient to accurately train the FA.

A third approach is to construct a training corpus from the PCFG by means of a (pseudo)random generator of sentences, such that sentences that are more likely according to the PCFG are generated with greater likelihood. This has been proposed by Jurafsky et al. (1994), for the special case of bigrams, extending a nonprobabilistic technique by Zue et al. (1991). It is not clear, however, whether this idea is feasible for training of finite-state models that are larger than bigrams. The reason is that very large corpora would have to be generated in order to obtain accurate probability estimates for the PFA. Note that the number of parameters of a bigram model is bounded by the square of the size of the lexicon; such a bound does not exist for general PFAs.

The current article discusses a fourth approach. In the limit, it is equivalent to the third approach above, as if an infinite corpus were constructed on which the PFA is trained, but we have found a way to avoid considering sentences individually. The key idea that allows us to handle an infinite set of strings generated by the PCFG is that we construct a new grammar that represents the intersection of the languages described by the input PCFG and the FA. Within this new grammar, we can compute the expected frequencies of transitions of the FA, using a fairly standard analysis of PCFGs. These expected frequencies then allow us to determine the assignment of probabilities to transitions of the FA that minimizes the KL distance between the PCFG and the resulting PFA.

1 In Nederhof (2000), several methods of approximation were discussed that lead to determinized approximating FAs that can be much larger than the input CFGs.

The only requirement is that the FA to be trained be unambiguous, by which we mean that each input string can be recognized by at most one computation of the FA. The special case of n-grams has already been formulated by Stolcke and Segal (1994), realizing an idea previously envisioned by Rimon and Herz (1991). An n-gram model is here seen as a (P)FA that contains exactly one state for each possible history of the $n-1$ previously read symbols. It is clear that such an FA is unambiguous (even deterministic) and that our technique therefore properly subsumes the technique by Stolcke and Segal (1994), although the way that the two techniques are formulated is rather different. Also note that the FA underlying an n-gram model accepts *any* input string over the alphabet, which does not hold for general (unambiguous) FAs.

Another application of our work involves determinization and minimization of PFAs. As shown by Mohri (1997), PFAs cannot always be determinized, and no practical algorithms are known to minimize arbitrary nondeterministic (P)FAs. This can be a problem when deterministic or small PFAs are required. We can, however, always compute a minimal deterministic FA equivalent to an input FA. The new results in this article offer a way to extend this determinized FA to a PFA such that it approximates the probability distribution described by the input PFA as well as possible, in terms of the KL distance.

Although the proposed technique has some limitations, in particular, that the model to be trained is unambiguous, it is by no means restricted to language models based on finite automata or context-free grammars, as several other probabilistic grammatical formalisms can be treated in a similar manner.

The structure of this article is as follows. We provide some preliminary definitions in Section 2. Section 3 discusses how the expected frequency of a rule in a PCFG can be computed. This is an auxiliary step in the algorithms to be discussed below. Section 4 defines a way to combine a PFA and a PCFG into a new PCFG that extends a well-known representation of the intersection of a regular and a context-free language. Thereby we merge the input model and the model to be trained into a single structure. This structure is the foundation for a number of algorithms, presented in section 5, which allow, respectively, training of an unambiguous FA on the basis of a PCFG (section 5.1), training of an unambiguous CFG on the basis of a PFA (section 5.2), and training of an unambiguous FA on the basis of a PFA (section 5.3).

2. Preliminaries

Many of the definitions on probabilistic context-free grammars are based on Santos (1972) and Booth and Thompson (1973), and the definitions on probabilistic finite automata are based on Paz (1971) and Starke (1972).

A **context-free grammar** $\mathcal{G}$ is a 4-tuple (Σ, N, S, R), where Σ and N are two finite disjoint sets of **terminals** and **nonterminals**, respectively, $S \in N$ is the **start symbol**, and R is a finite set of **rules**, each of the form $A \to \alpha$, where $A \in N$ and $\alpha \in (\Sigma \cup N)^*$. A **probabilistic context-free grammar** $\mathcal{G}$ is a 5-tuple $(\Sigma, N, S, R, p_{\mathcal{G}})$, where Σ, N, S and R are as above, and $p_{\mathcal{G}}$ is a function from rules in R to probabilities.

In what follows, symbol a ranges over the set Σ, symbols w, v range over the set Σ^*, symbols A, B range over the set N, symbol X ranges over the set $\Sigma \cup N$, symbols α, β, γ range over the set $(\Sigma \cup N)^*$, symbol ρ ranges over the set R, and symbols d, e range over the set R^*. With slight abuse of notation, we treat a rule $\rho = (A \to \alpha) \in R$ as an **atomic** symbol when it occurs within a string $d\rho e \in R^*$. The symbol ϵ denotes the empty string. String concatenation is represented by operator $\cdot$ or by empty space.

For a fixed (P)CFG $\mathcal{G}$, we define the relation $\Rightarrow$ on triples consisting of two strings $\alpha, \beta \in (\Sigma \cup N)^*$ and a rule $\rho \in R$ by $\alpha \overset{\rho}{\Rightarrow} \beta$, if and only if α is of the form $wA\delta$ and β is of the form $w\gamma\delta$, for some $w \in \Sigma^*$ and $\delta \in (\Sigma \cup N)^*$, and $\rho = (A \rightarrow \gamma)$. A **leftmost derivation** (in $\mathcal{G}$) is a string $d = \rho_1 \cdots \rho_m$, $m \geq 0$, such that $\alpha_0 \overset{\rho_1}{\Rightarrow} \alpha_1 \overset{\rho_2}{\Rightarrow} \cdots \overset{\rho_m}{\Rightarrow} \alpha_m$, for some $\alpha_0, \ldots, \alpha_m \in (\Sigma \cup N)^*$; $d = \epsilon$ is always a leftmost derivation. In the remainder of this article, we let the term *derivation* refer to leftmost derivation, unless specified otherwise. If $\alpha_0 \overset{\rho_1}{\Rightarrow} \cdots \overset{\rho_m}{\Rightarrow} \alpha_m$ for some $\alpha_0, \ldots, \alpha_m \in (\Sigma \cup N)^*$, then we say that $d = \rho_1 \cdots \rho_m$ **derives** α_m from α_0, and we write $\alpha_0 \overset{d}{\Rightarrow} \alpha_m$; ϵ derives any $\alpha_0 \in (\Sigma \cup N)^*$ from itself. A derivation d such that $S \overset{d}{\Rightarrow} w$, for some $w \in \Sigma^*$, is called a **complete** derivation. We say that $\mathcal{G}$ is **unambiguous** if for each $w \in \Sigma^*$, $S \overset{d}{\Rightarrow} w$ for at most one $d \in R^*$.

Let $\mathcal{G}$ be a fixed PCFG $(\Sigma, N, S, R, p_{\mathcal{G}})$. For $\alpha, \beta \in (\Sigma \cup N)^*$ and $d = \rho_1 \cdots \rho_m \in R^*$, $m \geq 0$, we define $p_{\mathcal{G}}(\alpha \overset{d}{\Rightarrow} \beta) = \prod_{i=1}^{m} p_{\mathcal{G}}(\rho_i)$ if $\alpha \overset{d}{\Rightarrow} \beta$, and $p_{\mathcal{G}}(\alpha \overset{d}{\Rightarrow} \beta) = 0$ otherwise. The probability $p_{\mathcal{G}}(w)$ of a string $w \in \Sigma^*$ is defined to be $\sum_d p_{\mathcal{G}}(S \overset{d}{\Rightarrow} w)$.

PCFG $\mathcal{G}$ is said to be **proper** if $\sum_{\rho,\alpha} p_{\mathcal{G}}(A \overset{\rho}{\Rightarrow} \alpha) = 1$ for all $A \in N$, that is, if the probabilities of all rules $\rho = (A \rightarrow \alpha)$ with left-hand side A sum to one. PCFG $\mathcal{G}$ is said to be **consistent** if $\sum_w p_{\mathcal{G}}(w) = 1$. Consistency implies that the PCFG defines a probability distribution on the set of terminal strings. There is a practical sufficient condition for consistency that is decidable (Booth and Thompson 1973).

A PCFG is said to be **reduced** if for each nonterminal A, there are $d_1, d_2 \in R^*$, $w_1, w_2 \in \Sigma^*$, and $\beta \in (\Sigma \cup N)^*$ such that $p_{\mathcal{G}}(S \overset{d_1}{\Rightarrow} w_1 A \beta) \cdot p_{\mathcal{G}}(w_1 A \beta \overset{d_2}{\Rightarrow} w_1 w_2) > 0$. In words, if a PCFG is reduced, then for each nonterminal A, there is at least one derivation $d_1 d_2$ with nonzero probability that derives a string $w_1 w_2$ from S and that includes some rule with left-hand side A. A PCFG $\mathcal{G}$ that is not reduced can be turned into one that is reduced and that describes the same probability distribution, provided that $\sum_w p_{\mathcal{G}}(w) > 0$. This **reduction** consists in removing from the grammar any nonterminal A for which the above conditions do not hold, together with any rule that contains such a nonterminal; see Aho and Ullman (1972) for reduction of CFGs, which is very similar.

A **finite automaton** $\mathcal{M}$ is a 5-tuple (Σ, Q, q_0, q_f, T), where Σ and Q are two finite sets of **terminals** and **states,** respectively, $q_0, q_f \in Q$ are the **initial** and **final** states, respectively, and T is a finite set of **transitions,** each of the form $r \overset{a}{\mapsto} s$, where $r \in Q - \{q_f\}, s \in Q$, and $a \in \Sigma$.[2] A **probabilistic finite automaton** $\mathcal{M}$ is a 6-tuple $(\Sigma, Q, q_0, q_f, T, p_{\mathcal{M}})$, where Σ, Q, q_0, q_f, and T are as above, and $p_{\mathcal{M}}$ is a function from transitions in T to probabilities.

In what follows, symbols q, r, s range over the set Q, symbol τ ranges over the set T, and symbol c ranges over the set T^*.

For a fixed (P)FA $\mathcal{M}$, we define a **configuration** to be an element of $Q \times \Sigma^*$, and we define the relation $\vdash$ on triples consisting of two configurations and a transition $\tau \in T$ by $(r, w) \overset{\tau}{\vdash} (s, w')$ if and only if w is of the form aw', for some $a \in \Sigma$, and $\tau = (r \overset{a}{\mapsto} s)$. A **computation** (in $\mathcal{M}$) is a string $c = \tau_1 \cdots \tau_m$, $m \geq 0$, such that $(r_0, w_0) \overset{\tau_1}{\vdash} (r_1, w_1) \overset{\tau_2}{\vdash} \cdots \overset{\tau_m}{\vdash} (r_m, w_m)$, for some $(r_0, w_0), \ldots, (r_m, w_m) \in Q \times \Sigma^*$; $c = \epsilon$ is always a computation. If $(r_0, w_0) \overset{\tau_1}{\vdash} \cdots \overset{\tau_m}{\vdash} (r_m, w_m)$ for some $(r_0, w_0), \ldots, (r_m, w_m) \in Q \times \Sigma^*$ and $c = \tau_1 \cdots \tau_m \in T^*$, then we write $(r_0, w_0) \overset{c}{\vdash} (r_m, w_m)$. We say that c **recognizes** w if $(q_0, w) \overset{c}{\vdash} (q_f, \epsilon)$.

2 That we only allow one final state is not a serious restriction with regard to the set of strings we can process; only when the empty string is to be recognized could this lead to difficulties. Lifting the restriction would encumber the presentation with treatment of additional cases without affecting, however, the validity of the main results.

Let $\mathcal{M}$ be a fixed FA (Σ, Q, q_0, q_f, T). The language $L(\mathcal{M})$ **accepted** by $\mathcal{M}$ is defined to be $\{w \in \Sigma^* \mid \exists_c[(q_0, w) \overset{c}{\vdash} (q_f, \epsilon)]\}$. We say $\mathcal{M}$ is **unambiguous** if for each $w \in \Sigma^*$, $(q_0, w) \overset{c}{\vdash} (q_f, \epsilon)$ for at most one $c \in T^*$. We say $\mathcal{M}$ is **deterministic** if for each $(r, w) \in Q \times \Sigma^*$, there is at most one combination of $\tau \in T$ and $(s, w') \in Q \times \Sigma^*$ such that $(r, w) \overset{\tau}{\vdash} (s, w')$. Turning a given FA into one that is deterministic and accepts the same language is called **determinization**. All FAs can be determinized. Turning a given (deterministic) FA into the smallest (deterministic) FA that accepts the same language is called **minimization**. There are effective algorithms for minimization of deterministic FAs.

Let $\mathcal{M}$ be a fixed PFA $(\Sigma, Q, q_0, q_f, T, p_{\mathcal{M}})$. For $(r, w), (s, v) \in Q \times \Sigma^*$ and $c = \tau_1 \cdots \tau_m \in T^*$, we define $p_{\mathcal{M}}((r, w) \overset{c}{\vdash} (s, v)) = \prod_{i=1}^{m} p_{\mathcal{M}}(\tau_i)$ if $(r, w) \overset{c}{\vdash} (s, v)$, and $p_{\mathcal{M}}((r, w) \overset{c}{\vdash} (s, v)) = 0$ otherwise. The probability $p_{\mathcal{M}}(w)$ of a string $w \in \Sigma^*$ is defined to be $\sum_c p_{\mathcal{M}}((q_0, w) \overset{c}{\vdash} (q_f, \epsilon))$.

PFA $\mathcal{M}$ is said to be proper if $\sum_{\tau, a, s:\ \tau = (r \overset{a}{\mapsto} s) \in T} p_{\mathcal{M}}(\tau) = 1$ for all $r \in Q - \{q_f\}$.

3. Expected Frequencies of Rules

Let $\mathcal{G}$ be a PCFG $(\Sigma, N, S, R, p_{\mathcal{G}})$. We assume without loss of generality that S does not occur in the right-hand side of any rule from R. For each rule ρ, we define

$$E(\rho) = \sum_{d, d', w} p_{\mathcal{G}}(S \overset{d \rho d'}{\Rightarrow} w) \tag{1}$$

If $\mathcal{G}$ is proper and consistent, (1) is the expected frequency of ρ in a complete derivation.

Each complete derivation $d\rho d'$ can be written as $d\rho d''d'''$, with $d' = d''d'''$, where

$$S \overset{d}{\Rightarrow} w'A\beta, A \overset{\rho}{\Rightarrow} \alpha, \alpha \overset{d''}{\Rightarrow} w'', \beta \overset{d'''}{\Rightarrow} w''' \tag{2}$$

for some A, α, β, w', w'', and w'''. Therefore

$$E(\rho) = \mathit{outer}(A) \cdot p_{\mathcal{G}}(\rho) \cdot \mathit{inner}(\alpha) \tag{3}$$

where we define

$$\mathit{outer}(A) = \sum_{d, w', \beta, d''', w'''} p_{\mathcal{G}}(S \overset{d}{\Rightarrow} w'A\beta) \cdot p_{\mathcal{G}}(\beta \overset{d'''}{\Rightarrow} w''') \tag{4}$$

$$\mathit{inner}(\alpha) = \sum_{d'', w''} p_{\mathcal{G}}(\alpha \overset{d''}{\Rightarrow} w'') \tag{5}$$

for each $A \in N$ and $\alpha \in (\Sigma \cup N)^*$. From the definition of *inner*, we can easily derive the following equations:

$$\mathit{inner}(a) = 1 \tag{6}$$

$$\mathit{inner}(A) = \sum_{\substack{\rho, \alpha: \\ \rho = (A \to \alpha)}} p_{\mathcal{G}}(\rho) \cdot \mathit{inner}(\alpha) \tag{7}$$

$$\mathit{inner}(X\beta) = \mathit{inner}(X) \cdot \mathit{inner}(\beta) \tag{8}$$

This can be taken as a recursive definition of *inner*, assuming $\beta \neq \epsilon$ in (8). Similarly, we can derive a recursive definition of *outer*:

$$outer(S) = 1 \tag{9}$$

$$outer(A) = \sum_{\substack{\rho, B, \alpha, \beta : \\ \rho = (B \rightarrow \alpha A \beta)}} outer(B) \cdot p_{\mathcal{G}}(\rho) \cdot inner(\alpha) \cdot inner(\beta) \tag{10}$$

for $A \neq S$.

In general, there may be cyclic dependencies in the equations for *inner* and *outer*; that is, for certain nonterminals A, *inner*(A) and *outer*(A) may be defined in terms of themselves. There may even be no closed-form expression for *inner*(A). However, one may approximate the solutions to arbitrary precision by means of fixed-point iteration.

4. Intersection of Context-Free and Regular Languages

We recall a construction from Bar-Hillel, Perles, and Shamir (1964) that computes the intersection of a context-free language and a regular language. The input consists of a CFG $\mathcal{G} = (\Sigma, N, S, R)$ and an FA $\mathcal{M} = (\Sigma, Q, q_0, q_f, T)$; note that we assume, without loss of generality, that $\mathcal{G}$ and $\mathcal{M}$ share the same set of terminals Σ.

The output of the construction is CFG $\mathcal{G}_\cap = (\Sigma, N_\cap, S_\cap, R_\cap)$, where $N_\cap = Q \times (\Sigma \cup N) \times Q$, $S_\cap = (q_0, S, q_f)$, and $R_\cap$ consists of the set of rules that is obtained as follows:

- For each rule $\rho = (A \rightarrow X_1 \cdots X_m) \in R$, $m \geq 0$, and each sequence of states $r_0, \ldots, r_m \in Q$, let the rule $\rho_\cap = ((r_0, A, r_m) \rightarrow (r_0, X_1, r_1) \cdots (r_{m-1}, X_m, r_m))$ be in $R_\cap$; for $m = 0$, $R_\cap$ contains a rule $\rho_\cap = ((r_0, A, r_0) \rightarrow \epsilon)$ for each state r_0.
- For each transition $\tau = (r \overset{a}{\mapsto} s) \in T$, let the rule $\rho_\cap = ((r, a, s) \rightarrow a)$ be in $R_\cap$.

Note that for each rule $(r_0, A, r_m) \rightarrow (r_0, X_1, r_1) \cdots (r_{m-1}, X_m, r_m)$ from $R_\cap$, there is a unique rule $A \rightarrow X_1 \cdots X_m$ from R from which it has been constructed by the above. Similarly, each rule $(r, a, s) \rightarrow a$ uniquely identifies a transition $r \overset{a}{\mapsto} s$. This means that if we take a derivation $d_\cap$ in $\mathcal{G}_\cap$, we can extract a sequence $h_1(d_\cap)$ of rules from $\mathcal{G}$ and a sequence $h_2(d_\cap)$ of transitions from $\mathcal{M}$, where h_1 and h_2 are string homomorphisms that we define pointwise as

$$h_1(\rho_\cap) = \rho \text{ if } \rho_\cap = ((r_0, A, r_m) \rightarrow (r_0, X_1, r_1) \cdots (r_{m-1}, X_m, r_m)) \tag{11}$$
$$\text{and } \rho = (A \rightarrow X_1 \cdots X_m)$$

$$\epsilon \text{ if } \rho_\cap = ((r, a, s) \rightarrow a) \tag{12}$$

$$h_2(\rho_\cap) = \tau \text{ if } \rho_\cap = ((r, a, s) \rightarrow a) \text{ and } \tau = (r \overset{a}{\mapsto} s) \tag{13}$$

$$\epsilon \text{ if } \rho_\cap = ((r_0, A, r_m) \rightarrow (r_0, X_1, r_1) \cdots (r_{m-1}, X_m, r_m)) \tag{14}$$

We define $h(d_\cap) = (h_1(d_\cap), h_2(d_\cap))$. It can be easily shown that if $h(d_\cap) = (d, c)$ and $S_\cap \overset{d_\cap}{\Rightarrow} w$, then for the same w, we have $S \overset{d}{\Rightarrow} w$ and $(q_0, w) \overset{c}{\vdash} (q_f, \epsilon)$. Conversely, if for some w, d, and c we have $S \overset{d}{\Rightarrow} w$ and $(q_0, w) \overset{c}{\vdash} (q_f, \epsilon)$, then there is precisely one derivation $d_\cap$ such that $h(d_\cap) = (d, c)$ and $S_\cap \overset{d_\cap}{\Rightarrow} w$.

It was observed by Lang (1994) that $\mathcal{G}_\cap$ can be seen as a **parse forest**, that is, a compact representation of all parse trees according to $\mathcal{G}$ that derive strings recognized by $\mathcal{M}$. The construction can be generalized to, for example, tree-adjoining grammars (Vijay-Shanker and Weir 1993) and range concatenation grammars (Boullier 2000; Bertsch and Nederhof 2001). The construction for the latter also has implications for linear context-free rewriting systems (Seki et al. 1991).

The construction has been extended by Nederhof and Satta (2003) to apply to a PCFG $\mathcal{G} = (\Sigma, N, S, R, p_\mathcal{G})$ and a PFA $\mathcal{M} = (\Sigma, Q, q_0, q_f, T, p_\mathcal{M})$. The output is a PCFG $\mathcal{G}_\cap = (\Sigma, N_\cap, S_\cap, R_\cap, p_\cap)$, where $N_\cap$, $S_\cap$, and $R_\cap$ are as before, and $p_\cap$ is defined by

$$p_\cap((r_0, A, r_m) \to (r_0, X_1, r_1) \cdots (r_{m-1}, X_m, r_m)) = p_\mathcal{G}(A \to X_1 \cdots X_m) \tag{15}$$

$$p_\cap((r, a, s) \to a) = p_\mathcal{M}(r \overset{a}{\mapsto} s) \tag{16}$$

If $d_\cap$, d, and c are such that $h(d_\cap) = (d, c)$, then clearly $p_\cap(d_\cap) = p_\mathcal{G}(d) \cdot p_\mathcal{M}(c)$.

5. Training Models on Models

We restrict ourselves to a few cases of the general technique of training a model on the basis of another model.

5.1 Training a PFA on a PCFG

Let us assume we have a proper and consistent PCFG $\mathcal{G} = (\Sigma, N, S, R, p_\mathcal{G})$ and an FA $\mathcal{M} = (\Sigma, Q, q_0, q_f, T)$ that is unambiguous. This FA may have resulted from (nonprobabilistic) approximation of CFG (Σ, N, S, R), but it may also be totally unrelated to $\mathcal{G}$. Note that an FA is guaranteed to be unambiguous if it is deterministic; any FA can be determinized. Our goal is now to assign probabilities to the transitions from FA $\mathcal{M}$ to obtain a proper PFA that approximates the probability distribution described by $\mathcal{G}$ as well as possible.

Let us define $\mathbb{1}$ as the function that maps each transition from T to one. This means that for each r, w, c and s, $\mathbb{1}((r, w) \overset{c}{\vdash} (s, \epsilon)) = 1$ if $(r, w) \overset{c}{\vdash} (s, \epsilon)$, and $\mathbb{1}((r, w) \overset{c}{\vdash} (s, \epsilon)) = 0$ otherwise.

Of the set of strings generated by $\mathcal{G}$, a subset is recognized by computations of $\mathcal{M}$; note again that there can be at most one such computation for each string. The expected frequency of a transition τ in such computations is given by

$$E(\tau) = \sum_{w, c, c'} p_\mathcal{G}(w) \cdot \mathbb{1}((q_0, w) \overset{c\tau c'}{\vdash} (q_f, \epsilon)) \tag{17}$$

Now we construct the PCFG $\mathcal{G}_\cap$ as explained in section 4 from the PCFG $\mathcal{G}$ and the PFA $(\Sigma, Q, q_0, q_f, T, \mathbb{1})$. Let $\tau = (r \overset{a}{\mapsto} s) \in T$ and $\rho = ((r, a, s) \to a)$. On the basis of the

properties of function h, we can now rewrite $E(\tau)$ as

$$\begin{aligned} E(\tau) &= \sum_{d,w,c,c'} p_{\mathcal{G}}(S \overset{d}{\Rightarrow} w) \cdot \mathbb{1}((q_0, w) \overset{c\tau c'}{\vdash} (q_f, \epsilon)) \\ &= \sum_{\substack{e,d,w,c,c': \\ h(e)=(d,c\tau c')}} p_{\mathcal{G}}(S \overset{d}{\Rightarrow} w) \cdot \mathbb{1}((q_0, w) \overset{c\tau c'}{\vdash} (q_f, \epsilon)) \\ &= \sum_{e,e',w} p_{\cap}(S_{\cap} \overset{e\rho e'}{\Rightarrow} w) \\ &= E(\rho) \end{aligned} \tag{18}$$

Hereby we have expressed the expected frequency of a transition $\tau = (r \overset{a}{\mapsto} s)$ in terms of the expected frequency of rule $\rho = ((r, a, s) \rightarrow a)$ in derivations in PCFG $\mathcal{G}_{\cap}$. It was explained in section 3 how such a value can be computed. Note that since by definition $\mathbb{1}(\tau) = 1$, also $p_{\cap}(\rho) = 1$. Furthermore, for the right-hand side a of ρ, $inner(a) = 1$. Therefore,

$$\begin{aligned} E(\tau) &= outer((r, a, s)) \cdot p_{\cap}(\rho) \cdot inner(a) \\ &= outer((r, a, s)) \end{aligned} \tag{19}$$

To obtain the required PFA $(\Sigma, Q, q_0, q_f, T, p_{\mathcal{M}})$, we now define the probability function $p_{\mathcal{M}}$ for each $\tau = (r \overset{a}{\mapsto} s) \in T$ as

$$p_{\mathcal{M}}(\tau) = \frac{outer((r, a, s))}{\sum_{a', s': (r \overset{a'}{\mapsto} s') \in T} outer((r, a', s'))} \tag{20}$$

That such a relative frequency estimator $p_{\mathcal{M}}$ minimizes the KL distance between $p_{\mathcal{G}}$ and $p_{\mathcal{M}}$ on the domain $L(\mathcal{M})$ is proven in the appendix.

An example with finite languages is given in Figure 1. We have, for example,

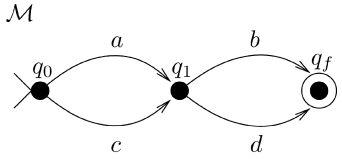
$$\begin{aligned} p_{\mathcal{M}}(q_0 \overset{a}{\mapsto} q_1) &= \frac{outer((q_0, a, q_1))}{outer((q_0, a, q_1)) + outer((q_0, c, q_1))} \\ &= \frac{\frac{1}{3}}{\frac{1}{3} + \frac{2}{3}} = \frac{1}{3} \end{aligned} \tag{21}$$

5.2 Training a PCFG on a PFA

Similarly to section 5.1, we now assume we have a proper PFA $\mathcal{M} = (\Sigma, Q, q_0, q_f, T, p_{\mathcal{M}})$ and a CFG $\mathcal{G} = (\Sigma, N, S, R)$ that is unambiguous. Our goal is to find a function $p_{\mathcal{G}}$ that lets proper and consistent PCFG $(\Sigma, N, S, R, p_{\mathcal{G}})$ approximate $\mathcal{M}$ as well as possible. Although CFGs used for natural language processing are usually ambiguous, there may be cases in other fields in which we may assume grammars are unambiguous.

$\mathcal{G}$

$S \to a\ b\ [1/3]$
$S \to c\ d\ [2/3]$



$\mathcal{G}_\cap$

$(q_0, S, q_f) \to (q_0, a, q_1)\ (q_1, b, q_f)\ [1/3]$
$(q_0, S, q_f) \to (q_0, c, q_1)\ (q_1, d, q_f)\ [2/3]$
$(q_0, a, q_1) \to a\ [1]$
$(q_1, b, q_f) \to b\ [1]$
$(q_0, c, q_1) \to c\ [1]$
$(q_1, d, q_f) \to d\ [1]$

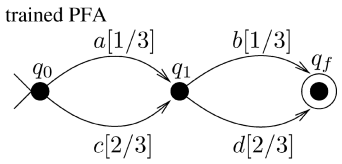


Figure 1
Example of input PCFG $\mathcal{G}$, with rule probabilities between square brackets, input FA $\mathcal{M}$, the reduced PCFG $\mathcal{G}_\cap$, and the resulting trained PFA.

Let us define $\mathbb{1}$ as the function that maps each rule from R to one. Of the set of strings recognized by $\mathcal{M}$, a subset can be derived in $\mathcal{G}$. The expected frequency of a rule ρ in those derivations is given by

$$E(\rho) = \sum_{d,d',w} p_{\mathcal{M}}(w) \cdot \mathbb{1}(S \overset{d\rho d'}{\Rightarrow} w) \tag{22}$$

Now we construct the PCFG $\mathcal{G}_\cap$ from the PCFG $\mathcal{G} = (\Sigma, N, S, R, \mathbb{1})$ and the PFA $\mathcal{M}$ as explained in section 4. Analogously to section 5.1, we obtain for each $\rho = (A \to X_1 \cdots X_m)$

$$\begin{aligned} E(\rho) &= \sum_{r_0,r_1,\ldots,r_m} E((r_0, A, r_m) \to (r_0, X_1, r_1) \cdots (r_{m-1}, X_m, r_m)) \\ &= \sum_{r_0,r_1,\ldots,r_m} \mathit{outer}((r_0, A, r_m)) \cdot \mathit{inner}((r_0, X_1, r_1) \cdots (r_{m-1}, X_m, r_m)) \end{aligned} \tag{23}$$

To obtain the required PCFG $(\Sigma, N, S, R, p_{\mathcal{G}})$, we now define the probability function $p_{\mathcal{G}}$ for each $\rho = (A \to \alpha)$ as

$$p_{\mathcal{G}}(\rho) = \frac{E(\rho)}{\sum_{\rho' = (A \to \alpha') \in R} E(\rho')} \tag{24}$$

The proof that this relative frequency estimator $p_{\mathcal{G}}$ minimizes the KL distance between $p_{\mathcal{M}}$ and $p_{\mathcal{G}}$ on the domain $L(\mathcal{G})$ is almost identical to the proof in the appendix for a similar claim from section 5.1.

5.3 Training a PFA on a PFA

We now assume we have a proper PFA $\mathcal{M}_1 = (\Sigma, Q_1, q_{0,1}, q_{f,1}, T_1, p_1)$ and an FA $\mathcal{M}_2 = (\Sigma, Q_2, q_{0,2}, q_{f,2}, T_2)$ that is unambiguous. Our goal is to find a function p_2 so that

proper PFA $(\Sigma, Q_2, q_{0,2}, q_{f,2}, T_2, p_2)$ approximates $\mathcal{M}_1$ as well as possible, minimizing the KL distance between p_1 and p_2 on the domain $L(\mathcal{M}_2)$.

One way to solve this problem is to map $\mathcal{M}_2$ to an equivalent right-linear CFG $\mathcal{G}$ and then to apply the algorithm from section 5.2. The obtained probability function $p_\mathcal{G}$ can be translated back to an appropriate function p_2. For this special case, the construction from section 4 can be simplified to the "cross-product" construction of finite automata (see, e.g., Aho and Ullman 1972). The simplified forms of the functions *inner* and *outer* from section 3 are commonly called *forward* and *backward*, respectively, and they are defined by systems of linear equations. As a result, we can compute exact solutions, as opposed to approximate solutions by iteration.

Appendix

We now prove that the choice of $p_\mathcal{M}$ in section 5.1 is such that it minimizes the Kullback-Leibler distance between $p_\mathcal{G}$ and $p_\mathcal{M}$, restricted to the domain $L(\mathcal{M})$. Without this restriction, the KL distance is given by

$$D(p_\mathcal{G} \| p_\mathcal{M}) = \sum_w p_\mathcal{G}(w) \cdot \log \frac{p_\mathcal{G}(w)}{p_\mathcal{M}(w)} \tag{25}$$

This can be used for many applications mentioned in section 1. For example, an FA $\mathcal{M}$ approximating a CFG $\mathcal{G}$ is guaranteed to be such that $L(\mathcal{M}) \supseteq L(\mathcal{G})$ in the case of most practical approximation algorithms. However, if there are strings w such that $w \notin L(\mathcal{M})$ and $p_\mathcal{G}(w) > 0$, then (25) is infinite, regardless of the choice of $p_\mathcal{M}$. We therefore restrict $p_\mathcal{G}$ to the domain $L(\mathcal{M})$ and normalize it to obtain

$$p_{\mathcal{G}|\mathcal{M}}(w) = \frac{p_\mathcal{G}(w)}{Z}, \text{ if } w \in L(\mathcal{M}) \tag{26}$$

$$0, \text{ otherwise} \tag{27}$$

where $Z = \sum_{w:w\in L(\mathcal{M})} p_\mathcal{G}(w)$. Note that $p_{\mathcal{G}|\mathcal{M}} = p_\mathcal{G}$ if $L(\mathcal{M}) \supseteq L(\mathcal{G})$. Our goal is now to show that our choice of $p_\mathcal{M}$ minimizes

$$\begin{aligned} D(p_{\mathcal{G}|\mathcal{M}} \| p_\mathcal{M}) &= \sum_{w:w\in L(\mathcal{M})} p_{\mathcal{G}|\mathcal{M}}(w) \cdot \log \frac{p_{\mathcal{G}|\mathcal{M}}(w)}{p_\mathcal{M}(w)} \\ &= \log \frac{1}{Z} + \frac{1}{Z} \sum_{w:w\in L(\mathcal{M})} p_\mathcal{G}(w) \cdot \log \frac{p_\mathcal{G}(w)}{p_\mathcal{M}(w)} \end{aligned} \tag{28}$$

As Z is independent of $p_\mathcal{M}$, it is sufficient to show that our choice of $p_\mathcal{M}$ minimizes

$$\sum_{w:w\in L(\mathcal{M})} p_\mathcal{G}(w) \cdot \log \frac{p_\mathcal{G}(w)}{p_\mathcal{M}(w)} \tag{29}$$

Now consider the expression

$$\prod_\tau p_\mathcal{M}(\tau)^{E(\tau)} \tag{30}$$

By the usual proof technique with Lagrange multipliers, it is easy to show that our choice of $p_{\mathcal{M}}$ in section 5.1, given by

$$p_{\mathcal{M}}(\tau) = \frac{E(\tau)}{\sum_{\tau', a', s' : \tau' = (r \overset{a'}{\mapsto} s') \in T} E(\tau')} \tag{31}$$

for each $\tau = (r \overset{a}{\mapsto} s) \in T$, is such that it maximizes (30), under the constraint of properness.

For $\tau \in T$ and $w \in \Sigma^*$, we define $\#_\tau(w)$ to be zero, if $w \notin L(\mathcal{M})$, and otherwise to be the number of occurrences of τ in the (unique) computation that recognizes w. Formally, $\#_\tau(w) = \sum_{c,c'} \mathbb{1}((q_0, w) \overset{c\tau c'}{\vdash} (q_f, \epsilon))$. We rewrite (30) as

$$\begin{aligned}
\prod_\tau p_{\mathcal{M}}(\tau)^{E(\tau)} &= \prod_\tau p_{\mathcal{M}}(\tau)^{\sum_w p_{\mathcal{G}}(w) \cdot \#_\tau(w)} \\
&= \prod_w \prod_\tau p_{\mathcal{M}}(\tau)^{p_{\mathcal{G}}(w) \cdot \#_\tau(w)} \\
&= \prod_w \left(\prod_\tau p_{\mathcal{M}}(\tau)^{\#_\tau(w)} \right)^{p_{\mathcal{G}}(w)} \\
&= \prod_{w : p_{\mathcal{M}}(w) > 0} p_{\mathcal{M}}(w)^{p_{\mathcal{G}}(w)} \\
&= \prod_{w : p_{\mathcal{M}}(w) > 0} 2^{p_{\mathcal{G}}(w) \cdot \log p_{\mathcal{M}}(w)} \\
&= \prod_{w : p_{\mathcal{M}}(w) > 0} 2^{p_{\mathcal{G}}(w) \cdot \log p_{\mathcal{M}}(w) - p_{\mathcal{G}}(w) \cdot \log p_{\mathcal{G}}(w) + p_{\mathcal{G}}(w) \cdot \log p_{\mathcal{G}}(w)} \\
&= \prod_{w : p_{\mathcal{M}}(w) > 0} 2^{-p_{\mathcal{G}}(w) \cdot \log \frac{p_{\mathcal{G}}(w)}{p_{\mathcal{M}}(w)} + p_{\mathcal{G}}(w) \cdot \log p_{\mathcal{G}}(w)} \\
&= 2^{-\sum_{w : p_{\mathcal{M}}(w) > 0} p_{\mathcal{G}}(w) \cdot \log \frac{p_{\mathcal{G}}(w)}{p_{\mathcal{M}}(w)}} \cdot 2^{\sum_{w : p_{\mathcal{M}}(w) > 0} p_{\mathcal{G}}(w) \cdot \log p_{\mathcal{G}}(w)}
\end{aligned} \tag{32}$$

We have already seen that the choice of $p_{\mathcal{M}}$ that maximizes (30) is given by (31), and (31) implies $p_{\mathcal{M}}(w) > 0$ for all w such that $w \in L(\mathcal{M})$ and $p_{\mathcal{G}}(w) > 0$. Since $p_{\mathcal{M}}(w) > 0$ is impossible for $w \notin L(\mathcal{M})$, the value of

$$2^{\sum_{w : p_{\mathcal{M}}(w) > 0} p_{\mathcal{G}}(w) \cdot \log p_{\mathcal{G}}(w)} \tag{33}$$

is determined solely by $p_{\mathcal{G}}$ and by the condition that $p_{\mathcal{M}}(w) > 0$ for all w such that $w \in L(\mathcal{M})$ and $p_{\mathcal{G}}(w) > 0$. This implies that (30) is maximized by choosing $p_{\mathcal{M}}$ such that

$$2^{-\sum_{w : p_{\mathcal{M}}(w) > 0} p_{\mathcal{G}}(w) \cdot \log \frac{p_{\mathcal{G}}(w)}{p_{\mathcal{M}}(w)}} \tag{34}$$

is maximized, or alternatively that

$$\sum_{w:p_{\mathcal{M}}(w)>0} p_{\mathcal{G}}(w) \cdot \log \frac{p_{\mathcal{G}}(w)}{p_{\mathcal{M}}(w)} \tag{35}$$

is minimized, under the constraint that $p_{\mathcal{M}}(w) > 0$ for all w such that $w \in L(\mathcal{M})$ and $p_{\mathcal{G}}(w) > 0$. For this choice of $p_{\mathcal{M}}$, (29) equals (35).

Conversely, if a choice of $p_{\mathcal{M}}$ minimizes (29), we may assume that $p_{\mathcal{M}}(w) > 0$ for all w such that $w \in L(\mathcal{M})$ and $p_{\mathcal{G}}(w) > 0$, since otherwise (29) is infinite. Again, for this choice of $p_{\mathcal{M}}$, (29) equals (35). It follows that the choice of $p_{\mathcal{M}}$ that minimizes (29) concurs with the choice of $p_{\mathcal{M}}$ that maximizes (30), which concludes our proof.

Acknowledgments

Comments by Khalil Sima'an, Giorgio Satta, Yuval Krymolowski, and anonymous reviewers are gratefully acknowledged. The author is supported by the PIONIER Project Algorithms for Linguistic Processing, funded by NWO (Dutch Organization for Scientific Research).

References

Aho, Alfred V. and Jeffrey D. Ullman. 1972. *Parsing*, volume 1 of *The Theory of Parsing, Translation and Compiling*. Prentice Hall, Englewood Cliffs, NJ.

Bar-Hillel, Yehoshua, M. Perles, and E. Shamir. 1964. On formal properties of simple phrase structure grammars. In Yehoshua Bar-Hillel, editor, *Language and Information: Selected Essays on Their Theory and Application*. Addison-Wesley, Reading, MA, pages 116–150.

Bertsch, Eberhard and Mark-Jan Nederhof. 2001. On the complexity of some extensions of RCG parsing. In *Proceedings of the Seventh International Workshop on Parsing Technologies*, pages 66–77, Beijing, October.

Booth, Taylor L. and Richard A. Thompson. 1973. Applying probabilistic measures to abstract languages. *IEEE Transactions on Computers*, C-22(5):442–450.

Boullier, Pierre. 2000. Range concatenation grammars. In *Proceedings of the Sixth International Workshop on Parsing Technologies*, pages 53–64, Trento, Italy, February.

Jurafsky, Daniel, Chuck Wooters, Gary Tajchman, Jonathan Segal, Andreas Stolcke, Eric Fosler, and Nelson Morgan. 1994. The Berkeley Restaurant Project. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP-94)*, pages 2139–2142, Yokohama, Japan.

Lang, Bernard. 1994. Recognition can be harder than parsing. *Computational Intelligence*, 10(4):486–494.

Manning, Christopher D. and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

Mohri, Mehryar. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

Mohri, Mehryar and Mark-Jan 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*. Kluwer Academic, pages 153–163.

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

Nederhof, Mark-Jan and Giorgio Satta. 2003. Probabilistic parsing as intersection. In *Proceedings of the Eighth International Workshop on Parsing Technologies*, pages 137–148, Laboratoire Lorrain de recherche en informatique et ses applications (LORIA), Nancy, France, April.

Paz, Azaria. 1971. *Introduction to Probabilistic Automata*. Academic Press, New York.

Rimon, Mori and J. Herz. 1991. The recognition capacity of local syntactic constraints. In *Proceedings of the Fifth Conference of the European Chapter of the ACL*, pages 155–160, Berlin, April.

Santos, Eugene S. 1972. Probabilistic grammars and automata. *Information and Control*, 21:27–47.

Seki, Hiroyuki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229.

Starke, Peter H. 1972. *Abstract Automata*. North-Holland, Amsterdam.

Stolcke, Andreas and Jonathan Segal. 1994. Precise *N*-gram probabilities from stochastic context-free grammars. In *Proceedings of the 32nd Annual Meeting of the ACL*, pages 74–79, Las Cruces, NM, June.

Vijay-Shanker, K. and David J. Weir. 1993. The use of shared forests in tree adjoining grammar parsing. In *Proceedings of the Sixth Conference of the European Chapter of the ACL*, pages 384–393, Utrecht, The Netherlands, April.

Zue, Victor, James Glass, David Goodine, Hong Leung, Michael Phillips, Joseph Polifroni, and Stephanie Seneff. 1991. Integration of speech recognition and natural language processing in the MIT Voyager system. In *Proceedings of the ICASSP-91*, Toronto, volume 1, pages 713–716.