Estimation of Generating Processes of Strings Represented with Patterns and Substitutions

Keisuke Otaki OOTAKI@IIP.IST.I.KYOTO-U.AC.JP
Akihiro Yamamoto AKIHIRO@I.KYOTO-U.AC.JP
Yoshida-Honmachi, Sakyo-ku, Kyoto, 606-8501, Japan.

Editors: Jeffrey Heinz, Colin de la Higuera and Tim Oates

Abstract

We formalize generating processes of strings based on patterns and substitutions, and give an algorithm to estimate a probability mass function on substitutions, which is an element of processes. Patterns are non-empty sequences of characters and variables. Variables indicate unknown substrings and are replaced with other patterns by substitutions. By introducing variables and substitutions, we can deal with the difficulty of preparing production rules in generative grammar and of representing context-sensitivity with them. Our key idea is to regard sequences of substitutions as generations of strings, and to give probabilities of substitutions like PCFG. In this study, after giving a problem to estimate a probability mass function from strings based on our formalization, we solve it by the Passing algorithm that runs in an iterative manner. Our experimental results with synthetic strings show that our method estimates probability mass functions with sufficient small errors.

Keywords: patterns, substitutions, latent structures of strings

1. Introduction

Our goal is to investigate a method to represent and find latent structures of strings using *patterns* and *substitutions* for applying them to the area of Knowledge Discovery from strings. *Patterns* are non-empty sequences of characters and variables. *Substitutions* are used for replacing variables in patterns with other patterns (Angluin, 1979; Shinohara and Arikawa, 1995). Our key strategy is to represent generations of strings as sequences of substitutions, and to regard latent structures of strings as their generating processes based on generations of strings represented with patterns and substitutions. During recent years, various data represented by strings such as genome sequences and documents in natural languages have been generated and stored. To find useful knowledge from them becomes an important task for Knowledge Discovery. For finding latent structures to understand properties of strings, *generative grammar*, in particular, *context free grammar* (CFG) have been adopted. However production rules of CFG often become complicated, and it is also difficult to represent *context-sensitivity* with them. In contrast, by introducing variables and substitutions, we can represent context-sensitivity. In this report, our task is to estimate a probability mass function, which is an element of our formalization from given strings. For that purpose, we give an algorithm that runs in an iterative manner. Our experimental results with synthetic strings show that our method works with sufficient small errors.

2. Preliminaries and Generating Processes

Let Σ be a finite and non-empty set of characters and X be an enumerable set of variables such that $\Sigma \cap X = \emptyset$. Variables are denoted with indices of natural numbers like x_0 and x_1. A *pattern* π is an element of $(\Sigma \cup X)^+ = P_{\Sigma,X}$. We can represent every pattern as $\pi = w_0 v_0 \cdots w_{n-1} v_{n-1} w_n$, where $w_i \in \Sigma^*$ $(0 \le i \le n)$ and $v_i \in X$ $(0 \le i \le n-1)$. For example, sequences $0x_01x_1x_2$, 011, and $x_0x_1x_0$ are patterns. We can represent every pattern as the *standard form* $\pi \downarrow = w_0 x_0 \cdots w_{n-1} x_{n-1} w_n$, where $w_i \in \Sigma^*$ $(0 \le i \le n)$ and $x_i \in X$ $(0 \le i \le n-1)$. Here we only deal with standard forms denoted by downarrows like $\pi \downarrow$. Refer the literature (Shinohara and Arikawa, 1995) for details.

Variables in patterns are replaced with other patterns by substitutions. A *substitution* θ is a finite set of the form $\{ v_1/\pi_1, \ldots, v_n/\pi_n \}$, where $v_i \in X$ and $\pi_i \in P_{\Sigma,X}$ for $1 \le i \le n$ and $\mathrm{dom}(\theta) = \{ v_i \mid v_i/\pi_i \in \theta \}$. If $n = 1$, we write the substitution without parenthesis as v_1/π_1. The new pattern $\pi\theta$ is the pattern obtained from π by simultaneously replacing each occurrence of v_i by π_i. We generate strings and patterns by substitutions chosen from a fixed set $\boldsymbol{\Theta}$ of substitutions, and they are characterized in the form of *refinement paths*. For two patterns π and τ, we define a refinement path from π to τ denoted by $\pi \Rightarrow_p \tau$, if there exists a sequence $p = (\theta_1, \ldots, \theta_n)$ such that $\theta_i \in \boldsymbol{\Theta}$ for $1 \le i \le n$ and $\pi\theta_1 \downarrow \ldots \theta_n \downarrow = \tau$. If $w \in \Sigma^+$, p is called a *generation* of w. We denote the set of all refinement paths from π to τ with a fixed set $\boldsymbol{\Theta}$ by $\mathit{Ref}_{\boldsymbol{\Theta}}(\pi, \tau)$.

Example 1 For a pattern $\pi = x_0$ and $\boldsymbol{\Theta} = \{ \theta_0, \theta_1, \theta_2, \theta_3 \}$ such that $\theta_0 = x_0/x_0x_0, \theta_1 = x_0/x_0x_1$, $\theta_2 = x_0/0$, and $\theta_3 = x_0/1$, it holds that $\pi\theta_1 \downarrow = x_0x_1$, $\pi\theta_1 \downarrow \theta_2 \downarrow = 0x_0$, and $\pi\theta_1 \downarrow \theta_2 \downarrow \theta_3 \downarrow = 01$. The sequence of $(\theta_1, \theta_2, \theta_3)$ is a refinement path from π to 01.

We regard refinement paths as generations of strings and treat generating processes that can be regarded as functions to provide refinement paths.

We define generating processes of strings and their probabilistic extension as systems and components, respectively. For the common initial pattern π and a fixed set $\boldsymbol{\Theta}$ of substitutions, we define a *system* as a pair $S = (\pi, \boldsymbol{\Theta})$. Just like in the case of context free languages, the language $L(S)$ of a system S is defined as $L(S) \equiv \{ w \in \Sigma^+ \mid \mathit{Ref}_{\boldsymbol{\Theta}}(\pi, w) \neq \emptyset \}$. We introduce a *probability mass function* μ on $\boldsymbol{\Theta}$ for characterizing refinement paths. We define a *component* c by a triple $c = (\pi, \boldsymbol{\Theta}, \mu)$, *i.e.*, a pair $c = (S, \mu)$, where $S = (\pi, \boldsymbol{\Theta})$ in which a probability of each substitution $\theta \in \boldsymbol{\Theta}$ is given by $\mu(\theta)$. When we correspond a system for a CFG, a component corresponds to a PCFG. We use $\mu(\boldsymbol{\Theta})$ to represent the set $\{ \theta_1 \colon \mu(\theta_1), \ldots, \theta_n \colon \mu(\theta_n) \}$ to represent the correspondence between θ_i and $\mu(\theta_i)$ for $\theta_i \in \boldsymbol{\Theta}$. We define the probability $\Pr_\mu(p)$ of a refinement path $p = (\theta_1, \ldots, \theta_n)$ on μ as $\Pr_\mu(p) = \prod_{i=1}^{n} \mu(\theta_i)$, and also define the *generation probability* $\Pr_g(w)$ of a string w as $\Pr_g(w) = \sum_{p \in \mathit{Ref}_{\boldsymbol{\Theta}}(\pi, w)} \Pr_\mu(p)$. Note that we assume that given sets of strings are represented as multisets with assuming that the occurrences of strings are related to the generation probabilities of them.

Example 2 Let an Extension component $c = (x_0, \boldsymbol{\Theta}, \mu)$ such that $\boldsymbol{\Theta} = \{ \theta_1, \theta_2, \theta_3, \theta_4 \}$, $\theta_0 = x_0/x_0x_0, \theta_1 = x_0/x_0x_1, \theta_2 = x_0/0, \theta_3 = x_0/1$, and $\mu(\boldsymbol{\Theta}) = \{ \theta_0 \colon 0.2, \theta_1 \colon 0.2, \theta_2 \colon 0.4, \theta_3 \colon 0.2 \}$. Strings 0 and 01 are generated by the component c with $p_1 = (\theta_2)$ and $p_2 = (\theta_1, \theta_2, \theta_3)$, respectively. The probability of them are $\Pr_\mu(p_1) = 0.4$ and $\Pr_\mu(p_2) = 0.016$ as follows:

$$x_0 \Rightarrow^{0.4}_{(\theta_2)} 0 \text{ and } x_0 \Rightarrow^{0.4}_{(\theta_1)} x_0x_1 \Rightarrow^{0.2}_{(\theta_2)} 0x_0 (= 0x_1 \downarrow) \Rightarrow^{0.2}_{(\theta_3)} 01,$$

where the number on each arrow shows $\mu(\theta)$ and θ is a applied substitution.

In this study, we consider *simple substitutions* and *left-most applications* instead of considering various substitutions. A substitution θ is *simple* if $\mathrm{dom}(\theta)$ is a singleton and it is either of the form of v/vu or v/c, where $v, u \in X$, and $c \in \Sigma$. We say that a substitution θ is applied to the left-most of a pattern π if $\mathrm{dom}(\theta) = \{ v \}$ and v is the left-most variable of π. A variable v in π is called *left-most* if no other variables occur in the left of v on π.

3. The Estimating Approach for the Probability Estimation Problem

We solve the Probability Estimation Problem, which is a problem to estimate $\mu(\mathbf{\Theta})$ from training data T and a system $S = (\pi, \mathbf{\Theta})$. In this report, we propose an heuristic algorithm called the *Passing algorithm* that runs in an iterative manner. It updates a hypothesis with difference and constants calculated with the current hypothesis and given data.

As a preliminary, we enumerate all refinement paths by constructing a search tree, where every edge represents a substitution and every node is labeled with a pattern. We can find all refinement paths of a string $w \in T$ by Depth-First Search (DFS) on the tree.

We introduce three types of *effects*, E_1, E_2, and E_3 of refinement paths. We calculate those effects for each path $p \in \mathit{Ref}_{\mathbf{\Theta}}(\pi, w)$ for $w \in T$. The first effect $E_1(p)$ is calculated on the hub h of p. We call the parent of the node labeled with w a *hub* h and the node labeled with a string w a *constant node*. If we have a path $p = (\theta_1, \dots, \theta_m) \in \mathit{Ref}_{\mathbf{\Theta}}(\pi, w)$, we increase the value $\mu(\theta_m)$. The value $\mathit{sign}(p, \theta_i) = 1.0$ if $\theta_m = \theta_i$ and -1.0 if $\theta_m \neq \theta_i$. The effect E_1 is defined as $E_1(p) = \{ \theta_i\colon \mathit{sign}(p, \theta_i) \mid 1 \leq i \leq |\mathbf{\Theta}| \}$.

The second one E_2 and the third one E_3 are defined with a refinement path from the root to the hub h and a path from the hub h to the constant node labeled with a string w, respectively. We define $r2h(p, \theta_i)$ and $r2d(p, \theta_i)$ respectively as the number of occurrences of θ_i on the refinement path p from the root to the hub h and that of the path p from the hub h to the node labeled with the string w. The effects E_2 and E_3 are defined as $E_2(p) = \{ \theta_i\colon r2h(p, \theta_i)/Z_2 \mid 1 \leq i \leq |\mathbf{\Theta}| \}$ and $E_3(p) = \{ \theta_i\colon r2d(p, \theta_i)/Z_3 \mid 1 \leq i \leq |\mathbf{\Theta}| \}$, where Z_2 and Z_3 are regularization constants.

We calculate a difference $\Delta(w)$ of a string w between the current hypothesis μ_{cur} and T for updating the hypothesis μ_{cur}. The difference is calculated by $\Delta(w) \equiv \Pr_g(w) - \Pr_E(w)$, where $\Pr_E(w) \equiv (\#\text{ of occurrences of } w \in T)/|T|$. The algorithm takes the three effects E_1, E_2 and E_3 multiplied with $\Delta(w)$ for $w \in T$ and some rates α, β, and γ such that $0 \leq \alpha, \beta, \gamma \leq 1$ into the hypothesis μ_{cur}. The initial values of $\mu(\mathbf{\Theta})$ is simply prepared as $(\frac{1}{|\mathbf{\Theta}|}, \dots, \frac{1}{|\mathbf{\Theta}|})$. This iteration is repeated until μ_{cur} is unchanged and the estimated result is denoted by $\hat{\mu}$. The above method, named the *Passing algorithm*, is illustrated in Figure 1.

4. Experiments

We prepared two components c_1 and c_2, that is, Duplication $c_1 = (x_0, \mathbf{\Theta}_1, \mu_1)$ such that $\mu_1(\mathbf{\Theta}_1) = \{ x_0/x_0x_0\colon 0.3, x_0/0\colon 0.15, x_0/1\colon 0.4, x_0/2\colon 0.15 \}$ and the Extension component c_2 which we used in Example 2. In experiments, we generate 100 strings using c_1 and c_2, and estimate $\mu(\mathbf{\Theta})$ with the parameter vector $(0.01, 0.01, 0.1)$ as (α, β, γ). Because of correspondences between components and PCFG, we can apply the *maximum likelihood estimation* (MLE) method by referring to the literature (Lari and Young, 1990). We will explain

```
Passing algorithm with an initial pattern $\pi$ and a set $\mathbf{\Theta}$ of substitutions
Input:  Training data $T$, Constants $M, \delta, \alpha, \beta, \gamma$
Output:  Estimated values $\hat{\mu}(\theta)$ for $\theta \in \mathbf{\Theta}$.
Procedure:
  For $i = 1$ to $M$ until $w$ exists such that $\Delta(w) < \delta$:
    0. Prepare initial values of $\mu_{cur}(\mathbf{\Theta})$ as $(\frac{1}{|\mathbf{\Theta}|}, \ldots, \frac{1}{|\mathbf{\Theta}|})$.
    For each $w \in T$:
      1. Prepare the DFS tree to find $Ref_{\mathbf{\Theta}}(\pi, w)$ and calculate $\Delta(w)$.
      For each $p \in Ref_{\mathbf{\Theta}}(\pi, w)$:
        2. Calculate three effects $E_1(p)$, $E_2(p)$ and $E_3(p)$.
        3. For $\theta \in \mathbf{\Theta}$, $\mu_{cur}(\theta) \leftarrow \mu_{cur}(\theta) + \Delta(w)(\alpha E_1(p)(\theta) + \beta E_2(p)(\theta) + \gamma E_3(p)(\theta))$.
```

Figure 1: The outline of the Passing algorithm.

Table 1: Estimated values $\hat{\mu}(\mathbf{\Theta})$ for c_1 and c_2 with two methods.

(a) Duplication: $c_1 = (x_0, \mathbf{\Theta}_1, \mu_1)$

Methods	$\hat{\mu}_1(x_0/x_0x_0)$	$\hat{\mu}_1(x_0/0)$	$\hat{\mu}_1(x_0/1)$	$\hat{\mu}_1(x_0/2)$	# of iters
Passing algorithm	0.230	0.190	0.400	0.180	50
MLE method	0.288	0.160	0.407	0.145	–

(b) Extension: $c_2 = (x_0, \mathbf{\Theta}_2, \mu_2)$

(α, β, γ)	$\hat{\mu}_2(x_0/x_0x_0)$	$\hat{\mu}_2(x_0/x_0x_1)$	$\hat{\mu}_2(x_0/0)$	$\hat{\mu}_2(x_0/1)$	# of iters
Passing algorithm	0.172	0.199	0.413	0.217	48
MLE method	0.169	0.385	0.171	0.275	–

the MLE method in Appendix A formally. The estimated results are listed in Tables 1(a) and 1(b). From the experimental results, the MLE method works better for c_1 because we do not need any heuristics for c_1, where we estimate the values from visible data. However, the simple MLE method does not work well for c_2 comparing with the Passing algorithm, in particular, for the value $\mu(\theta_1)$. We conclude that our algorithm can estimate the values $\mu(\mathbf{\Theta})$ well as seen in Tables 1(a) and 1(b). The substitutions θ_0 and θ_1 concern structures of strings, and therefore, we conclude that the results of the Passing algorithm are better. We conjecture that it is because of the difference between our formalization and PCFG.

5. Conclusion and Future Work

We formalized generating processes of strings based on patterns and substitutions, and proposed a method of estimating probability mass functions of the processes. Our experimental results with synthetic data shows that our method works well compared with the results for the MLE method. In our future work, we investigate theoretical properties of our algorithm more precisely and improve our methods by referring to the method for PCFG like EM algorithm and Bayesian inference. We can simulate generations of strings by estimated values, therefore we will try to find some features of given strings by using the estimated values. For example, we would like to apply our formalization to find frequent substrings based on generating processes. We also consider to try other experiments for real data. which are represented by strings and codes like binary codes in order to examine the properties of our formalization.

References

D. Angluin. Finding patterns common to a set of strings (extended abstract). In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 130–141, 1979.

K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.

T. Shinohara and S. Arikawa. Pattern inference. In *GOSLER Final Report*, pages 259–291, 1995.

Appendix A. The MLE Method for PCFG and Correspondences

The MLE method for our problems based on patterns and substitutions are given as follows. Let $\mathbf{p} = \{ p_1, \ldots, p_N \}$ be a set of refinement paths which generate a given set of strings $T = \{ w_1, \ldots, w_N \}$. We denote the likelihood $L(\mu)$ as follows:

$$L(\mu) = \prod_{i=1}^{N} \mathrm{Pr}_\mu(p_i) = \prod_{\theta \in \mathbf{\Theta}} \mu(\theta)^{f_\theta(\mathbf{p})},$$

where $f_\theta(\mathbf{p})$ is the number of occurrences of the substitution θ in the set of refinement paths $\mathbf{p}$. For $v \in X$, we need to satisfy

$$\sum_{\theta \in \{\phi \in \mathbf{\Theta} | \mathrm{dom}(\phi) = v\}} \mu(\theta) = 1.$$

On this restriction, we introduce Lagrange constants. In our settings, where we only adopt left-most applications and simple substitutions, we only consider a Lagrange constant c and maximize the following function:

$$\Lambda(\mu, c) = L(\mu) - c \left(\sum_{\theta \in \mathbf{\Theta}} \mu(\theta) - 1 \right).$$

By setting $\frac{\partial \Lambda}{\partial \mu(\theta)}$ to 0, we can estimate the probability $\mu(\theta)$ by calculating *relative frequencies* as

$$\hat{\mu}(\theta) = \frac{f_\theta(\mathbf{p})}{\sum_{\theta \in \mathbf{\Theta}} f_\theta(\mathbf{p})}.$$

We can use this method for the component c_1 because for each generated string w with c_1, it holds that $|Ref_{\mathbf{\Theta}_1}(\pi_1, w)| = 1$, that is, they have unique refinement paths and each refinement path corresponds to each string.

For the component c_2, we need some adjustments for estimating $\mu(\mathbf{\Theta}_2)$ from generated strings $T = \{ w_1, \ldots, w_N \}$ because the refinement paths for generated strings with c_2 are not unique. For this problem, we calculate *weighted estimated frequencies* as follows:

$$\hat{\mu}(\theta) = \frac{\sum_{p \in Ref_{\mathbf{\Theta}}(\pi, w)} f_\theta(p) \mathrm{Pr}_\mu(p)}{\sum_{p \in Ref_{\mathbf{\Theta}}(\pi, w)} \sum_{\phi \in \mathbf{\Theta}} f_\phi(p) \mathrm{Pr}_\mu(p)}.$$

Note that we can regard the Passing algorithm as a heuristic extension of the MLE method. Updating the hypothesis with effects E_2 and E_3 in the Passing algorithm can be regarded as that they have the same function of counting the number of occurrences of each $\theta \in \mathbf{\Theta}$ in refinement paths with some constants by regularization. We introduced the effect E_1 with aiming at estimating better results for the substitution x_0/x_0x_0 or x_0/x_0x_1 because such substitutions determine the forms of generated strings by variables. It can be regarded as important invisible features of strings how many times such substitutions are applied in our settings.