Discovering Repetitive Expressions and Affinities from Anthologies of Classical Japanese Poems

Koichiro Yamamoto[1], Masayuki Takeda[1,2], Ayumi Shinohara[1], Tomoko Fukuda[3], and Ichirō Nanri[3]

[1] Department of Informatics, Kyushu University 33, Fukuoka 812-8581, Japan
[2] PRESTO, Japan Science and Technology Corporation (JST)
[3] Junshin Women's Junior College, Fukuoka 815-0036, Japan
{k-yama, takeda, ayumi}@i.kyushu-u.ac.jp
{tomoko-f@muc, nanri-i@msj}.biglobe.ne.jp

Abstract. The class of pattern languages was introduced by Angluin (1980), and a lot of studies have been undertaken on it from the theoretical viewpoint of learnabilities. However, there have been few practical studies except for the one by Shinohara (1982), in which patterns are restricted so that every variable occurs at most once. In this paper, we distinguish *repetitive* variables from those occurring only once within a pattern, and focus on the number of occurrences of a repetitive-variable and the length of strings it matches, in order to model the rhetorical device based on repetition of words in classical Japanese poems. Preliminary result suggests that it will lead to characterization of individual anthology, which has never been achieved, up till now.

1 Introduction

Recently, we have tackled several problems in analyzing classical Japanese poems, Waka. In [12], we successfully discovered from Waka poems characteristic patterns, named Fushi, which are read-once patterns whose constant parts are restricted to sequences of auxiliary verbs and postpositional particles. In [10], we addressed the problem of semi-automatically finding similar poems, and discovered unheeded instances of Honkadori (poetic allusion), one important rhetorical device in Waka poems based on specific allusion to earlier famous poems. On the contrary, we in [11] succeeded to discover expression highlighting differences between two anthologies by two closely related poets (e.g., master poet and disciples). In the present paper, we focus on *repetition*.

Repetition is the basis for many poetic forms. The use of repetition can heighten the emotional impact of a piece. This device, however, has received little attentions in the case of Waka poetry. One of the main reasons might be that a Waka poem takes a form of short poem, namely, it consists only of five lines and thirty-one syllables, arranged 5-7-5-7-7, and therefore the use of repetition is often considered to waste words (letters) under this tight limitation. In fact, some poets/scholars in earlier times taught their disciples *never* to repeat a word in a Waka poem. They considered word repetition as 'disease' to be avoided. This device, however, gives a remarkable effect if skillfully used, even in Waka poetry. The following poem, composed by priest Egyō (lived in the latter half of the 10th-century), is a good example of repetition, where two words 'nawo' and 'kiku' are respectively used twice [1].

> HA-SHI-NO-NA-WO/NA-WO-U-TA-TA-NE-TO/KI-KU-HI-TO-NO/
> KI-KU-HA-MA-KO-TO-KA/U-TSU-TSU-NA-GA-RA-NI (EGYŌ-SHŪ #195)

Since there has been few studies on this poetic device in the long research history of Waka poetry, it is necessary to develop a method of automatically extracting (candidates for) instances of the repetition from database. To retrieve instances of repetition like above, we consider the pattern matching problem for patterns such as $\star x \star x \star y \star y \star$, where $\star$ is the *variable-length don't care* (VLDC), a wildcard that matches any strings, and x, y are variables that match any non-empty strings.

Recall the *pattern languages* proposed by Angluin [2]. A *pattern* is a string in $\Pi = (\Sigma \cup V)^+$, where V is an infinite set $\{x_1, x_2, \dots\}$ of variables and $\Sigma \cap V = \emptyset$. For example, $ax_1bx_2x_1$ is a pattern, where $a, b \in \Sigma$. The *language* of a pattern π is the set of strings obtained by replacing variables in π by non-empty strings. For example, $L(ax_1bx_2x_1) = \{aubvu \mid u, v \in \Sigma^+\}$.

Although the membership problem is NP-complete for the class of Angluin patterns as shown in [2], it becomes polynomial-time solvable when the number of variables occurring within π is bounded by a fixed number k. Several subclasses have been investigated from the viewpoint of polynomial-time learnability. For example, the classes of *read-once patterns* (every variable occurs only at once) and *one-variable patterns* (only one variable is contained) are known to be polynomial-time learnable [2]. In the present paper, we try to study subclasses from viewpoints of pattern matching and similarity computation.

It should be mentioned that the class of *regular expressions with back referencing* [1] is considered as a superclass of the Angluin patterns. The membership for this class is also known to be NP-complete.

On the other hand, we attempted in [10] to semi-automatically discover similar poems from an accumulation of about 450,000 Waka poems in a machine-readable form. As mentioned above, one of the aims was to discover unheeded instances of Honkadori. The method is simple: Arrange all possible pairs of poems in decreasing order of their similarities, and then scholarly scrutinize a first part. The key to success in this approach is how to develop an appropriate similarity measure. Traditionally, the scheme of weighted edit distance with a weight matrix may have been used to quantify affinities between strings. This scheme, however, requires a fine tuning of quadratically many weights in a matrix with the alphabet size, by a hand-coding or a heuristic criterion. As an alternative idea, we introduced a new framework called *string resemblance systems* (SRSs

[1] We inserted the hyphens '-' between syllables, each of which was written as one Kana character although romanized here. One can see that every syllable consists of either a single vowel or a consonant and a vowel. Thus there can be no consonantal clusters and every syllable ends in one of the five vowels *a, i, u, e, o*.

for short) [10]. In this framework, similarity of two strings is evaluated via a pattern that matches both of them, with the support by an appropriate function that associates the quantity of resemblance candidate patterns. This scheme bridges a gap between optimal pattern discovery (see, e.g., [5]) and similarity computation.

An SRS is specified by (1) a *pattern set* to which common patterns belong, and (2) a *pattern score function* that maps each pattern in the set to the quantity of resemblance. For example, if we choose the set of patterns with VLDCs and define the score of a pattern to be the number of symbols in it, then the obtained measure is the length of the longest common subsequence (LCS) of two strings. In fact, the strings `acdeba` and `abdac` have a common pattern $\mathtt{a}{\star}\mathtt{d}{\star}\mathtt{a}{\star}$ which contains three symbols.

With this framework one can easily design and modify his/her measures. In fact we designed some measures as combinations of pattern set and pattern score function along with the framework, and reported successful results in discovering unnoticed instances of Honkadori [10]. The discovered affinities raised an interesting issue for Waka studies, and we could give a convincing conclusion to it:

1. We have proved that one of the most important poems by Fujiwara-no-Kanesuke, one of the renowned thirty-six poets, was in fact based on a model poem found in Kokin-Shū. The same poem had been interpreted just to show "frank utterance of parents' care for their child." Our study revealed the poet's techniques in composition half hidden by the heart-warming feature of the poem by extracting the same structure between the two poems[2].
2. We have compared Tametada-Shū, the mysterious anthology unidentified in Japanese literary history, with a number of private anthologies edited after the middle of the Kamakura period (the 13th-century) using the same method, and found that there are about 10 pairs of similar poems between Tametada-Shū and Sōkon-Shū, an anthology by Shōtetsu. The result suggests that the mysterious anthology was edited by a poet in the early Muromachi period (the 15th-century). There have been surmised dispute about the editing date since one scholar suggested the middle of Kamakura period as a probable one. We have had a strong evidence about this problem.

In this paper, we focus on the class of Angluin patterns and on its subclasses, and discuss the problems of the pattern-matching, the similarity computation, and the pattern discovery. It should be emphasized that although many studies has been undertaken to the class of Angluin patterns and its subclasses, most of them has been done from the theoretical viewpoint of learnability. The only exception is due to Shinohara [9]. He mentioned practical applications, but they are limited to the subclass called the read-once patterns (referred to as regular patterns in [9]). We show in this paper the first practical application of Angluin

[2] *Asahi*, one of Japan's leading newspapers, made a front-page report of this discovery (26 May, 2001).

patterns that are not limited to the read-once patterns. As our framework quantifies similarities between strings by weighting patterns common to the strings, we modify the definition of patterns as follows:

– Substitute a gap symbol $\star$ for every variable occurring only once in a pattern.
– Associate each variable x with an integer $\mu(x)$ so that the variable x matches a string w only if the length of w is at least $\mu(x)$. (In the original setting in [2], $\mu(x) = 1$ for all variable x.)

Since we are interested only in repetitive strings in a Waka poem, there is no need to name non-repetitive strings. It suffices to use gap symbols $\star$ instead of variables for representing non-repetitive strings. Thus, the first item is rather for the sake of simplification. On the contrary, the second item is an essential augmentation by which the score of a pattern π can be sensitive to the values of $\mu(x)$ for variables x in π. In fact, we are strongly interested in the length of repeated string when analyzing repetitive expressions in Waka poems.

Fig. 1 is an instance of Honkadori we discovered in [10]. The two poems have several common expressions, such as, "na-ka-ra-he-te" and "to-shi-so-he-ni-ke-ru." One can notice that both the poems use the repetition of words. Namely, the Kokin-Shū poem and the Shin-Kokin-Shū repeat "nakara" (stem of verb "nagarafu"; name of a bridge) and "matsu" (wait; pine tree), respectively. This strengthens the affinities based on existence of common substrings.

Poem alluded to. (Kokin-Shū #826) Sakanoue-no-Korenori.

A-FU-KO-TO-WO	*Without seeing you,*
NA-KA-RA-NO-HA-SHI-NO	*I have lived on*
NA-KA-RA-HE-TE	*Adoring you ever*
KO-HI-WA-TA-RU-MA-NI	*Like the ancient bridge of Nagara*
TO-SHI-SO-HE-NI-KE-RU	*And many years have passed on.*

Allusive-variation. (Shin-Kokin-Shū #1636) Nijoin Sanuki.

NA-KA-RA-HE-TE	*Like the ancient pine tree of longevity*
NA-HO-KI-MI-KA-YO-WO	*On the mount of expectation called "Matsuyama,"*
MA-TSU-YA-MA-NO	*I have lived on*
MA-TSU-TO-SE-SHI-MA-NI	*Expecting your everlasting reign*
TO-SHI-SO-HE-NI-KE-RU	*And many years have passed on.*

Fig. 1. Discovered instance of poetic allusion.

It may be relevant to mention that this work is a multidisciplinary study between the literature and the computer science. In fact, the second author from the last is a Waka researcher and the last author is a linguist in Japanese language.

2 A Uniform Framework for String Similarity

This section briefly sketches the framework of string resemblance systems according to [10]. Gusfield [6] pointed out that in dealing with string similarity the language of alignments is often more convenient than the language of edit operations. Our framework is a generalization of the alignment based scheme and is based on the notion of *common patterns.*

Before describing our scheme, we need to introduce some notation. The set of strings over an alphabet Σ is denoted by Σ^*. The length of a string u is denoted by $|u|$. The string of length 0 is called the *empty string*, and denoted by ε. Let $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. Let us denote by $\boldsymbol{R}$ the set of real numbers. A *pattern system* is a triple of a finite alphabet Σ, a set Π of descriptions called *patterns*, and a function L that maps a pattern in Π to a subset of Σ^*. $L(\pi)$ is called the *language* of a pattern $\pi \in \Pi$. A pattern $\pi \in \Pi$ *match* a string $w \in \Sigma^*$ if w belongs to $L(\pi)$. A pattern π in Π is a *common pattern* of strings w_1 and w_2 in Σ^* if π matches both of them.

Definition 1. *A* string resemblance system *(SRS) is a 4-tuple $\langle \Sigma, \Pi, L, score\rangle$, where $\langle \Sigma, \Pi, L\rangle$ is a pattern system and score is a pattern score function that maps a pattern in Π to a real number.*

The *similarity* $\mathrm{SIM}(x, y)$ between strings x and y with respect to $\langle \Sigma, \Pi, L, score\rangle$ is defined by $\mathrm{SIM}(x, y) = \max\{score(\pi) \mid \pi \in \Pi \text{ and } x, y \in L(\pi)\}$. When the set $\{score(\pi) \mid \pi \in \Pi \text{ and } x, y \in L(\pi)\}$ is empty or the maximum does not exist, SIM(x,y) is undefined.

The above definition regards similarity computation as *optimal pattern discovery.* Our framework thus bridges a gap between similarity computation and pattern discovery. In [10], we defined the homomorphic SRSs and showed that the class of homomorphic SRSs covers most of the known similarity (dissimilarity) measures, such as, the edit distance, the weighted edit distance, the Hamming distance, the LCS measure. We also extended in [10] this class to the semi-homomorphic SRSs, and the similarity measures we developed in [8] for musical sequence comparison fall into this class.

We can handle a variety of string (dis)similarity by changing the pattern system and the pattern score function. The pattern systems appearing in the above examples are, however, restricted to homomorphic ones. Here, we shall mention SRSs with non-homomorphic pattern systems An *order-free pattern* (or *fragmentary pattern*) is a multiset $\{u_1, \ldots, u_k\}$ such that $k > 0$ and $u_1, \ldots, u_k \in \Sigma^+$, and is denoted by $\pi[u_1, \ldots, u_k]$. The language of pattern $\pi[u_1, \ldots, u_k]$ is the set of strings that contain the strings $u_1, \ldots, u_k$ without overlaps. The membership problem of the order-free patterns is NP-complete [7], and the similarity computation is NP-hard in general as shown in [7]. However, the membership problem is polynomial-time solvable when k is fixed. The class of order-free patterns plays an important role in finding similar poems from anthologies of Waka poems [10].

The pattern languages, introduced by Angluin [2], is also interesting for our framework.

Definition 2 (Angluin pattern system). *The Angluin pattern system is a pattern system $\langle \Sigma, (\Sigma \cup V)^+, L\rangle$, where V is an infinite set $\{x_1, x_2, \ldots\}$ of variables with $\Sigma \cap V = \emptyset$, and $L(\pi)$ is the set of strings $\pi \cdot \theta$ such that θ is a homomorphism from $(\Sigma \cup V)^+$ to Σ^+ such that $c \cdot \theta = c$ for every $c \in \Sigma$.*

In this paper we discuss SRSs with the Angluin pattern system.

3 Computational Complexity

Definition 3. Membership Problem for pattern system $\langle \Sigma, \Pi, L \rangle$.
Given a pattern $\pi \in \Pi$ *and a string* $w \in \Sigma^*$*, determine whether or not* $w \in L(\pi)$.

Theorem 1 ([2]). Membership problem for Angluin pattern system *is NP-complete.*

Definition 4. Similarity Computation with respect to SRS $\langle \Sigma, \Pi, L, \mathit{score} \rangle$. *Given two strings* $w_1, w_2 \in \Sigma^*$*, find a pattern* $\pi \in \Pi$ *with* $\{w_1, w_2\} \subseteq L(\pi)$ *that maximizes* $\mathit{score}(\pi)$.

Theorem 2. *For an SRS with Angluin pattern system,* Similarity Computation *is NP-hard in general.*

Proof. We consider the following problem, that is a decision version of a special case of Similarity Computation with $w_1 = w_2$, and show its NP-completeness. Optimal Pattern with respect to SRS $\langle \Sigma, \Pi, L, \mathit{score} \rangle$: Given a string $w \in \Sigma^*$ and an integer k, determine whether or not there is a pattern $\pi \in \Pi$ such that $w \in L(\pi)$ and $\mathit{score}(\pi) \geq k$.

We give a reduction from Membership Problem for Angluin pattern system $\langle \Sigma, \Pi, L \rangle$ to Optimal Pattern with respect to SRS with Angluin pattern system $\langle \Sigma', \Pi', L', \mathit{score} \rangle$ for a specific score function *score* defined as follows. Let $\Sigma' = \Sigma \cup \{\#\}$ with $\# \notin \Sigma$. We take a one-to-one mapping $\langle \cdot \rangle$ from $\Pi' = (\Sigma \cup V)^+$ to Σ^* that is log-space computable with respect to $|\pi|$. We define the score function $\mathit{score} : \Pi' \to \boldsymbol{R}$ by $\mathit{score}(\pi') = 1$ if π' is of the form $\pi' = \pi\#\langle \pi \rangle$ for some $\pi \in \Pi = (\Sigma \cup V)^+$, and $\mathit{score}(\pi') = 0$ otherwise.

For a given instance $\pi \in \Pi$ and $w \in \Sigma^*$ of Membership Problem for Angluin pattern system, let us consider $w' = w\#\langle \pi \rangle$ and $k = 1$ as an input to Optimal Pattern. Then we can see that there is a pattern $\pi' \in \Pi'$ with $w' \in L(\pi')$ and $\mathit{score}(\pi') = 1$ if and only if $w \in L(\pi)$, since $w' \in L(\pi')$ if and only if $\pi' = \pi\#\langle \pi \rangle$ and $w \in L(\pi)$. This completes the proof. $\square$

4 Practical Aspects

Recall that similarities between strings are quantified by weighting patterns common to them in our framework. For a finer weighting, we augment the descriptive power of Angluin patterns by putting a restriction on the length of a string matched by each variable. Namely, we associate each variable x with an integer $\mu(x)$ such that the variable x matches a string w only if $\mu(x) \leq |w|$. For example, suppose that $\pi_1 = z_1 x z_2 x z_3$ and $\pi_2 = z_1 y z_2 y z_3$, where $\mu(x) = 2$, $\mu(y) = 3$, and $\mu(z_1) = \mu(z_2) = \mu(z_3) = 0$. Then, π_1 is common to the strings *bcaaabbaac* and *acabbaabbbb*, but π_2 is not. This enables us to define a score function so that it is sensitive to the lengths of strings substituted for variables.

On the other hand, as we have seen in the last section, similarity computation as well as membership problem is intractable in general for Angluin pattern system. From a practical point of view, it is valuable to consider subclasses of the pattern system that are tractable.

Let $occ_x(\pi)$ denote the number of occurrences of a variable x within a pattern $\pi \in (\Sigma \cup V)^+$. For example, $occ_x(abxcyxbz) = 2$. A variable x is said to be *repetitive* w.r.t. π if $occ_x(\pi) > 1$. A pattern π is said to be *read-once* if π contains no repetitive variables. Historically, read-once patterns are called *regular patterns* because the induced languages are regular [9]. The membership problem of the read-once patterns is solvable in linear time. A *k-repetitive-variable pattern* is a pattern that has at most k repetitive-variables. It is not difficult to see that:

Theorem 3. *The membership problem of the k-repetitive-variable patterns can be solved in $O(n^{2k+1})$ time for input of size n.*

That is, non-repetitive variables do not matter. Moreover, we are interested only in repeated strings in text strings. For these reasons, we substitute $\star$ for each of the non-repetitive variables in a pattern. Patterns are then strings over $(\Sigma \cup V \cup \{\star\})$, in which every variable is repetitive. For example the above pattern $abxcyxbz$ is written as $abxc \star xb\star$.

Despite the polynomial-time computability, the membership problem of the k-repetitive-variable patterns requires much time to solve. The similarity computation is therefore very slow in practice. For this reason, we in this paper restrict ourselves to the case of $k = 1$, namely, the one-repetitive-variable patterns. In order to efficiently solve the membership problem and similarity computation for this class, we utilize a kind of filtering technique. For example, when the pattern $a \star xxb \star cx$ matches a string w, then the candidate strings for substituting for x must occur at least three times in w *without overlaps.* We obtain such substring statistics on a given string w by exploiting such data structures as the *minimal augmented suffix trees* developed by Apostolico and Preparata [3,4].

Suffix tree [6] for a string w is a tree structure that represents all suffices of w as paths from the root to leaves, so that every node except leaves have at least two children. Suffix trees are useful for the task of various string processing [6]. Each node v corresponds to a substring $\tilde{v}$ of w. For each internal node v, we associate the number of leaves of the subtree rooted at v. It corresponds to the number of (possibly overlapped) occurrences $\tilde{v}$ in w to the node (see Fig. 2 (a)).

Minimal augmented suffix tree is an augmented version of the suffix tree, where additional nodes are introduced to count non-overlapping occurrences. (see Fig. 2 (b)).

5 Application to Waka Data

In this section, we present and discuss the results of our experiments carried out on *the Eight Imperial Anthologies*, the first eight of the imperial anthologies compiled by emperor commands, listed in Table 1.

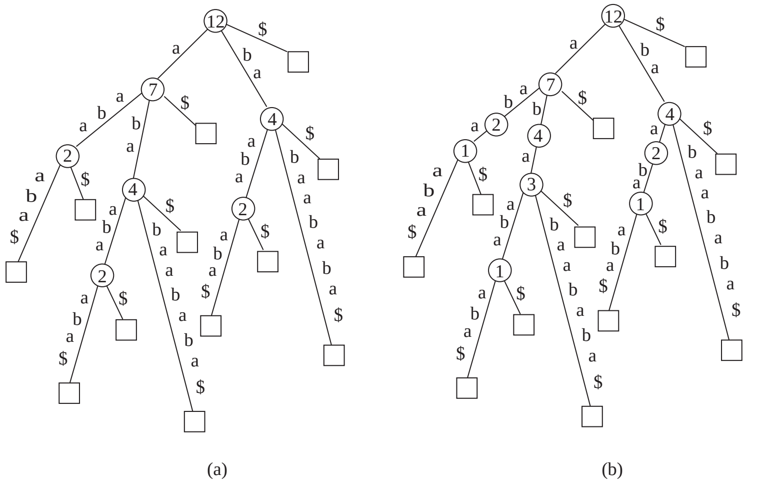

Fig. 2. (a)Suffix tree and (b)minimal augmented suffix tree for string *ababaababa*\$. The number associated to each internal node denotes the number occurrences of the string in the string, where occurrence means *possibly overlapped* occurrence in (a) and *non-overlapped* occurrence in (b). For example, the string *aba* occurs four times in the string *ababaababa*, but it appears only three times without overlapping.

Table 1. Eight Imperial Anthologies.

no.	anthology	compilation	# poems
I	Kokin-Shū	905	1,111
II	Gosen-Shū	955–958	1,425
III	Shūi-Shū	1005–1006	1,360
IV	Go-Shūi-Shū	1087	1,229
V	Kinyō-Shū	1127	717
VI	Shika-Shū	1151	420
VII	Senzai-Shū	1188	1,290
VIII	Shin-Kokin-Shū	1216	2,005

5.1 Similarity Computation

For a success in discovery, we want to put an appropriate restriction on the pattern system and on the pattern score function by using some domain knowledge. However, there are few studies on repetition of words in Waka poems as stated before, and therefore we do not in advance know what kind of restriction is effective.

We take a stepwise-refinement approach, namely, we start with very simple pattern system and score function, and then improve them based on analysis of obtained results. Here we restrict ourselves to one-repetitive-variable patterns. Moreover, we use a simple pattern score function that is not sensitive to characters or VLDCs in the patterns. Namely, the score of $a\star xxb\star cx$ is identical to that of $\star x\star x\star x\star$, for example. Despite this simplification, we wish to pay attention to how long the strings that match variable x are. Thus, a one-repetitive-variable pattern π is essentially expressed as two integers: $occ_x(\pi)$ and $\mu(x)$. We assume that the score function is non-decreasing with respect to $occ_x(\pi)$ and to $\mu(x)$.

We compared the anthology Kokin-Shū with two anthologies Gosen-Shū and Shin-Kokin-Shū. The score function we used is defined by $score(\pi) = occ_x(\pi) \cdot \mu(x)$. The frequency distributions are shown in Table 2. From the table, there seem relatively higher similarities between Kokin-Shū and Gosen-Shū, compared with Kokin-Shū and Shin-Kokin-Shū. We examined a first part of a list of poem pairs arranged in the decreasing order of similarity value. However, we had impressions that most of pairs with high similarity value are dissimilar, probably because the pattern system we used is too simple to quantify the affinities concerning repetition techniques. See the poems shown in Fig. 3. All the poems are matched by the pattern $\star x \star x \star$ with $\mu(x) = 4$. The first three poems are similar each other, while the other pairs are dissimilar. It seems that information about the locations at which a string occurs repeatedly is important.

Table 2. Frequency distribution on similarity values in comparison of Kokin-Shū with Gosen-Shū and Shin-Kokin-Shū. Note that similarity values cannot be 1, 2, 3, 5, 7 because of the definition of the pattern score function. The frequencies for any similarity values not present here are all 0.

	0	4	6	8	10
Gosen-Shū	1,390,030	178,331	1,944	37	8
Shin-Kokin-Shū	1,962,550	244,776	2,173	11	0

KA-SU-KA-NO-HA/KE-FU-HA-NA-YA-KI-SO/WA-KA-KU-SA-NO/
TSU-MA-<u>MO-KO-MO-RE-RI</u>/WA-RE-<u>MO-KO-MO-RE-RI</u>/ (KOKIN-SHŪ #17)

TO-SHI-NO-U-CHI-NI/HA-RU-HA-KI-NI-KE-RI/HI-TO-TO-SE-WO/
KO-SO-<u>TO-YA-I-HA-MU</u>/KO-TO-SHI-<u>TO-YA-I-HA-MU</u>/ (KOKIN-SHŪ #1)

HI-RU-NA-RE-YA/MI-SO-MA-KA-HE-TSU-RU/TSU-KI-KA-KE-WO/
KE-FU-<u>TO-YA-I-HA-MU</u>/KI-NO-FU-<u>TO-YA-I-HA-MU</u>/ (GOSEN-SHŪ #1100)

HA-RU-KA-SU-MI/TA-TE-RU-YA-I-TSU-KO/MI-<u>YO-SHI-NO-NO</u>/
<u>YO-SHI-NO-NO</u>-YA-MA-NI/YU-KI-HA-FU-RI-TSU-TSU/ (KOKIN-SHŪ #3)

<u>TSU-RA-KA-RA</u>-HA/O-NA-SHI-KO-KO-RO-NI/<u>TSU-RA-KA-RA</u>-M/
TSU-RE-NA-KI-HI-TO-WO/KO-HI-M-TO-MO-SE-SU/ (GOSEN-SHŪ #592)

Fig. 3. Poems that are matched by the same pattern $\star x \star x \star$ with $\mu(x) = 4$. All pairs have a unique similarity value. The first three poems can be considered to 'share' the same poetic device and are closely similar, while some pairs are dissimilar.

Moreover, we observed that there are a lot of meaningless repetitions of strings, especially when $\mu(x)$ is relatively small, say, $\mu(x) = 2$. It seems better to restrict ourselves to repetition of strings occurring at the beginning or the end of a line in order to remove such repetitions.

We assume the lines of a poem are parenthesized by $[,]$. Then, the pattern $[\star][x\star][x\star][\star][\star]$, for example, matches any poem whose second and third lines begin with a same string. We want to use the set of such patterns as the pattern set, but the number of such patterns is $3^5 = 243$, which makes the similarity computation impractical. However, by using the Minimal Augmented Suffix Trees, we can filter out a wasteful computation and perform the computation in reasonable time. The results are shown in Table 3. By examining a first part, we confirmed that this time pairs with a high similarity value are closely similar.

Table 3. Improved results. Frequency distribution on similarity values in comparison of Kokin-Shū with Gosen-Shū and Shin-Kokin-Shū. Note that similarity values cannot be 1, 2, 3, 5, 7 because of the definition of the pattern score function. The frequencies for any similarity values not present here are all 0.

	0	4	6	8	10
Gosen-Shū	1,569,925	407	14	1	3
Shin-Kokin-Shū	2,208,888	583	39	0	0

5.2 Characterization of Anthologies

Table 4 shows the most 30 patterns occurring in Kokin-Shū. The table illustrates variations of word repetition techniques.

Table 4. Most frequent 30 patterns in Kokin-Shū.

freq.	pattern	freq.	pattern	freq.	pattern
11	$[\star][\star][x\star][x\star][\star]$	3	$[\star x][\star][\star x][\star][\star]$	1	$[x\star][\star][x\star][\star][\star x]$
10	$[x\star][x\star][\star][\star][\star]$	3	$[\star][x\star][\star][\star][x\star]$	1	$[x\star][\star][\star][\star][\star x]$
10	$[\star][x\star][x\star][\star][\star]$	3	$[\star][\star x][\star][\star x][\star]$	1	$[\star x][\star][x\star][\star][\star]$
7	$[x\star][\star][\star][\star][x\star]$	3	$[\star][\star][x\star][\star][x\star]$	1	$[\star x][\star][\star][\star][\star x]$
5	$[\star][\star x][\star][\star][\star x]$	3	$[\star][\star][\star][\star x][\star x]$	1	$[\star][x\star][\star x][\star][\star]$
5	$[\star][\star][\star x][x\star][\star]$	2	$[x\star][\star][x\star][\star][\star]$	1	$[\star][x\star][\star][x\star][\star]$
5	$[\star][\star][\star][x\star][x\star]$	2	$[\star x][\star][\star][\star x][\star]$	1	$[\star][\star x][x\star][\star][\star]$
4	$[x\star][\star][\star][x\star][\star]$	2	$[\star x][\star][\star][\star][x\star]$	1	$[\star][\star][x\star][\star x][\star]$
4	$[\star x][x\star][\star][\star][\star]$	2	$[\star][\star x][\star x][\star][\star]$	1	$[\star][\star][x\star][\star][\star x]$
4	$[\star][\star][\star x][\star x][\star]$	1	$[x\star][x\star][\star][\star][x\star]$	0	$[x\star][x\star][x\star][x\star][x\star]$

For every pattern of the above mentioned form, we collected the poems that are matched by it from the first eight imperial anthologies shown in Table 1. The results are summarized in Table 5. The first four anthologies have a

Table 5. Characterization of anthologies. I, II, III, IV, V, VI, VII, VIII represent Kokin-Shū, Gosen-Shū, Shūi-Shū, Go-Shūi-Shū, Kinyō-Shū, Shika-Shū, Senzai-Shū, Shin-Kokin-Shū, respectively,

$(occ_x(\pi), \mu(x))$	I	II	III	IV	V	VI	VII	VIII
$(2,2)$	96	104	118	108	24	22	77	112
$(2,3)$	23	20	28	31	5	9	17	19
$(2,4)$	10	7	13	5	4	5	3	1
$(2,5)$	5	5	10	3	2	2	1	0
$(3,2)$	2	11	2	3	0	1	1	0
$(3,3)$	0	0	0	2	0	1	0	0
$(3,4)$	0	0	0	0	0	0	0	0
$(3,5)$	0	0	0	0	0	0	0	0
$(4,2)$	0	5	0	0	0	0	0	0
$(4,3)$	0	0	0	0	0	0	0	0
$(4,4)$	0	0	0	0	0	0	0	0
$(4,5)$	0	0	0	0	0	0	0	0
$(5,2)$	0	1	0	0	0	0	0	0
$(5,3)$	0	0	0	0	0	0	0	0
$(5,4)$	0	0	0	0	0	0	0	0
$(5,5)$	0	0	0	0	0	0	0	0

considerable amount of poems that use repetition of words, even for a large value of $\mu(x)$. This is contrasted with Shin-Kokin-Shū where limited to a small value of $\mu(x)$. This might be a reflection of the editor's preferences or of literary trend. Anyway, pursuing the reason for such differences will provide clues for further investigation on literary trend or the editors' personalities.

6 Concluding Remarks

The Angluin pattern language has been studied mainly from theoretical viewpoints. There are no practical applications except for those limited to the read-once patterns. This paper presented the first practical application of the Angluin pattern languages that are not limited to read-once patterns. We hope that pattern matching and similarity computation for the patterns discussed in this paper possibly lead to discovering overlooked aspects of individual poets.

We distinguished repetitive variables (i.e., occurring more than once in a pattern) from non-repetitive variables, and associated each variable x with an integer $\mu(x)$ as the lower bound to the length of strings the variable x matches. This enables us to give a pattern score depending upon the lengths of strings substituted for variables. For one-repetitive-variable pattern, we presented a way of speed-up of pattern matching, which uses substring statistics from minimal augmented suffix tree of a given string as a filter that excludes patterns which cannot match it. Preliminary experiment showed this idea successfully speeds up the pattern matching against many patterns repeatedly.

In this paper, we restricted ourselves to one-repetitive-variable patterns and to repetition of words which occur at the beginning or the end of lines of Waka poem. The restriction played an important role but we want to consider a slightly more complex patterns. For example, the following two poems are matched by the pattern $[\star][\star][x\star][xx\star][\star]$.

[SHI-RA-YU-KI-NO][YA-HE-FU-RI-SHI-KE-RU][$\underline{\text{KA-HE-RU}}$-YA-MA]
[$\underline{\text{KA-HE-RU}}$-$\underline{\text{KA-HE-RU}}$-MO][O-I-NI-KE-RU-KA-NA] (KOKIN-SHŪ #902)

[A-FU-KO-TO-HA][MA-HA-RA-NI-A-ME-RU][$\underline{\text{I-YO}}$-SU-TA-RE]
[$\underline{\text{I-YO}}$-$\underline{\text{I-YO}}$-WA-RE-WO][WA-HI-SA-SU-RU-KA-NA] (SHIKA-SHŪ #244)

Moreover, the next poem is matched by the pattern $[x\star][y\star][x*][x*][y*]$ that contains two-repetitive-variables.

[$\underline{\text{WA-SU-RE}}$-SHI-TO][$\underline{\text{I}}$-HI-TSU-RU-NA-KA-HA][$\underline{\text{WA-SU-RE}}$-KE-RI]
[$\underline{\text{WA-SU-RE}}$-MU-TO-KO-SO][$\underline{\text{I}}$-FU-HE-KA-RI-KE-RE] (GO-SHŪI-SHŪ #886)

To deal with more general patterns like these ones will be future work.

References

1. A. V. Aho. *Handbook of Theoretical Computer Science*, volume A, Algorithm and Complexity, chapter 5, pages 255–295. Elsevier, Amsterdam, 1990.
2. D. Angluin. Finding patterns common to a set of strings. *J. Comput. Sys. Sci.*, 21:46–62, 1980.
3. A. Apostolico and F. Preparata. Structural properties of the string statistics problem. *J. Comput. & Syst. Sci.*, 31(3):394–411, 1985.
4. A. Apostolico and F. Preparata. Data structures and algorithms for the string statistics problem. *Algorithmica*, 15(5):481–494, 1996.
5. H. Arimura. Text data mining with optimized pattern discovery. In *Proc. 17th Workshop on Machine Intelligence*, Cambridge, July 2000.
6. D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, New York, 1997.
7. H. Hori, S. Shimozono, M. Takeda, and A. Shinohara. Fragmentary pattern matching: Complexity, algorithms and applications for analyzing classic literary works. In *Proc. 12th Annual International Symposium on Algorithms and Computation (ISAAC'01)*, 2001. To appear.
8. T. Kadota, M. Hirao, A. Ishino, M. Takeda, A. Shinohara, and F. Matsuo. Musical sequence comparison for melodic and rhythmic similarities. In *Proc. 8th International Symposium on String Processing and Information Retrieval (SPIRE2001).* IEEE Computer Society, 2001. To appear.
9. T. Shinohara. Polynomial-time inference of pattern languages and its applications. In *Proc. 7th IBM Symp. Math. Found. Comp. Sci.*, pages 191–209, 1982.

10. M. Takeda, T. Fukuda, I. Nanri, M. Yamasaki, and K. Tamari. Discovering instances of poetic allusion from anthologies of classical Japanese poems. *Theor. Comput. Sci.* To appear.
11. M. Takeda, T. Matsumoto, T. Fukuda, and I. Nanri. Discovering characteristic expressions from literary works. *Theor. Comput. Sci.* To appear.
12. M. Yamasaki, M. Takeda, T. Fukuda, and I. Nanri. Discovering characteristic patterns from collections of classical Japanese poems. *New Gener. Comput.*, 18(1):61–73, 2000.