PATTERN DISCOVERY OF GENOME SEQUENCES BY SUBSTRING AMPLIFICATION

Ikeda, Daisuke
Computing and Communications Center,Kyushu University

Hirokawa, Sachio
Computing and Communications Center,Kyushu University

Yamada, Yasuhiro
Department of Informatics,Kyushu University

出版情報：Proceedings of International Symposium on Information Science and Electrical Engineering. 2003, pp.637-640, 2003-11
バージョン：
権利関係：

PATTERN DISCOVERY OF GENOME SEQUENCES BY SUBSTRING AMPLIFICATION

Daisuke Ikeda, Sachio Hirokawa

Computing and Communications Center,
Kyushu University

Yasuhiro Yamada

Department of Informatics,
Kyushu University

ABSTRACT

In this paper, we study a problem which is, given a set of genome sequences, to find common subsequences. We assume that the sequences are generated by some fixed but unknown pattern. The authors developed a method, called "substring amplification," to find the template part of a pattern from semi-structured documents, such as HTML files, generated by the pattern. Substring amplification exploits the disparity of frequency distributions between the template and background parts, and so requires only positive data. In HTML files, many characters are used and the length of a successive part of a template is enough long compared to genome sequences. In this paper, we examine the applicability of the method to genome sequences in which a constant sequence is embedded. By a series of experiments in which the length and alphabet size of the embedded sequences are varied, we show the effectiveness and limit of our method to genome sequences.

1. INTRODUCTION

It has been a basic and important problem in computer science to find a pattern common to given strings. One of such problems is the longest common subsequence problem. Other similar problems are considered in many fields: the longest common substring problem and maximal repetitions in stringology [1], frozen phrase and named entity extraction in natural language processing [2], pattern language learning [3, 4, 5], contents extraction by template discovery in Web mining [6, 7], and so on. Another related problem is finding frequent patterns and it has been well studied in data/text mining [8, 9, 10]. Cache and compression algorithms also find frequent segments of an input stream in an online manner.

Another research field in which finding common or frequent patterns is important is genome informatics. Target data in this field are huge amount of sequences which consist of small number of alphabets. In biologic evolution, sequences have been changed (nearly) randomly. On the other hand, it is assumed that important biologic functions are preserved among invariant segments of sequences. Therefore, the problem is important in this field.

It has been well studied to find common parts among input sequences using various approach and techniques in this field: machine learning, Bayesian networks, EM algorithm, sampling, hidden Markov model, Fourier transform, and so on. In general, it takes too many computational resources to find common patterns.

Complexity required to find common patterns heavily depends on the format of patterns. For example, the longest common substring is solved linearly but the longest common subsequence problem (or multiple alignment with no gap penalty) is NP-complete [11]. In this paper, we consider a set of substrings as a pattern. Note that a pattern is not a sequence of substrings, which is known as a subsequence pattern. In other words, the considered problem is to find local alignment from multiple sequence iteratively without knowledge about the length of common subsequences. This is a part of motif discovery. Thus, this is important in spite of the simplicity of the pattern's format.

We formally defined the template discovery problem as follows [12]: An input of the problem is a set of strings generated by some fixed but unknown pattern in the context of the pattern language [3]. The problem is to find the constant part of an unknown pattern from a finite set of positive strings. But we do no consider correspondence of variables. This is same as the machine learning of regular languages [5]. The learnability of the pattern language is difficult if the pattern is restricted to be regular. Instead that such restrictions on variable occurrence, we assume that substituted strings are generated by some natural distribution. By the natural distribution, we mean that (*) a distribution which gives a small possibility for a long string. On the other hand, the constant part is obviously common to all input strings. The key idea is that there exists a clear disparity between frequency distributions of substrings in the constant part and the one in the substituted string part. We call our approach "substring amplification."

We experimentally showed that if any constant string has some adequate length, we can find such a string by substring amplification [12, 13], and HTML files generated by some template meet the condition (*) because (1) a template for HTML files is enough long, (2) many characters are used in HTML files and (3) it is rarely happened that the

same long substrings occur many times in contents (variable parts). On the other hand, the alphabet size of genome sequences are quite small and the length of motif or common parts considered in the literature are relatively short. Therefore, we study applicability of our approach to genome sequences.

2. PATTERN LANGUAGES

Let Σ be an finite alphabet and $V = \{x_1, x_2, \ldots, \}$ be an infinite set of *variable*. A *pattern* is a string over $\Sigma \cup V$. A pattern is *regular* if each variable in the pattern appears at most once. In the sequel, we assume that all patterns are regular.

A word generated by a pattern is one obtained by replacing all variables by some strings over Σ. A set of all words generated by some pattern p is denoted by $L(p)$.

A finite set $S \subset \Sigma^*$ is called a *sample*. Given a sample S, a pattern p is *descriptive* with respect to S if $S \subseteq L(p)$ and there dose not exist $p' \neq p$ such that $L(p') \subset L(p)$.

Angluin introduced the pattern language and studied its learnability [3]. The problem is, given a sample S, to find a descriptive pattern with respect to S.

3. SUBSTRING AMPLIFICATION

Many learning algorithms consider patterns in which occurrence of variables is restricted [3, 4, 5] since the problem for general patterns is intractable. However, our interest is not to find variable parts but to find the template part, that is, identifying constant substrings of a given pattern. Therefore, we consider another restriction on the pattern.

Consider a pattern $p = w_1 x_1 \cdots w_m x_m$ such that $w_i \in \Sigma^*$ and $x_i \in V$ for each $1 \leq i \leq m$. The *template* of p is $(w_1, \ldots, w_m)$ and denoted by $t(p)$. The *width* of the template is denoted by $|t(p)|$ and defined as follows: $|t(p)| = \min\{|w_i| \mid w_i \in t(p)\}$.

Our goal is to find substrings common to given strings. In other words, we want to find the template. This is the same as the goal of the machine learning of regular pattern language [5] because if a regular pattern is found then we can extract the template from the pattern, and conversely, if a template is found then we can construct a regular pattern by inserting variables.

The learnability of the pattern language is difficult if the pattern is restricted to be regular. Therefore, we developed another approach using frequency distributions [12, 13]. We defined formally the problem as follows [12]:

Definition 1 *Given a sample S which is generated by unknown pattern $p = w_1 x_1 \cdots w_m x_m$ with width k according to some natural distribution $D(x_1, \ldots, x_m)$, the* template discovery problem *is to find the template $t = (w_1, \ldots, w_m)$ of p which is descriptive with respect to S.*

Note that the probabilistic distribution D is unknown.

Non-template parts (background) of strings in S do not have regularity because some natural distributions are assumed. Therefore, any substring in the background does not appear frequently. On the other hand, any substrings of the template must appear frequently because many of the sample S contains the template even if noise data are included. This is the key idea of substring amplification.

Given a sample S, the proposed method just counts all substrings in S. Then, for each frequency f, it outputs pairs $(f, V(f))$, where the number $V(f)$ of different substrings whose frequency are exactly f. In other words, there are $V(f)$ substrings which appear exactly f times in S. $V(f)$ is known as the *frequency of frequency* of Zipf's law. Finally, it draws a graph, called a *frequency graph*, which shows $F(f)(= V(f) * f)$ for all f in log scale. We call $F(f)$ the *amplified frequency*.

When a natural text such as a novel is given, $F(f)$ is drastically and smoothly decreasing as f is increasing (see Fig. 2). This phenomena is famous as the Zipf's law.

But, when m strings are given and they share a template, there must be clear peaks in the frequency graph. This is because common substrings in the template appear at least m times (if the template completely appears in all input strings) and so $F(f)$ is at most $n^2 * m$ for $f = m$, where n is the length of the template and n^2 is maximal number of substrings in the template. This $F(f)$ is a relatively higher since the frequency distribution for the variable part is natural and it is rarely for an adequately long substring to appear frequently in the variable part due to the assumption (*). So, we can extract substrings which constitute peaks as common patterns.

Substring amplification utilizes the frequency distribution. There are many other algorithms to do in a similar way. In most of such algorithms, the length n to count substrings is fixed. In fact, bigram ($n = 2$) or trigram ($n = 3$) is usually used in natural language processing. On the other hand, the proposed algorithm does not fix the length. Moreover the algorithm sums up frequencies for all n (in our implementation the maximal n is fixed by some constant). We call this approach "substring amplification" which makes the difference among the distributions clear.

Substring amplification receives only positive strings. This contrasts to many other existing algorithms using both positive and negative (or background) strings [14, 15]. Our approach is supported by the fact that it works well for HTML files generated for some template in our previous experiments [12]. Fig. 1 is a frequency graph for Sankei Shimbun. where S is a set of 50 HTML files collected from Sankei Shimbun[1]. We can see a clear peak at $f = 50$, which

[1] `http://www.sankei.co.jp/`

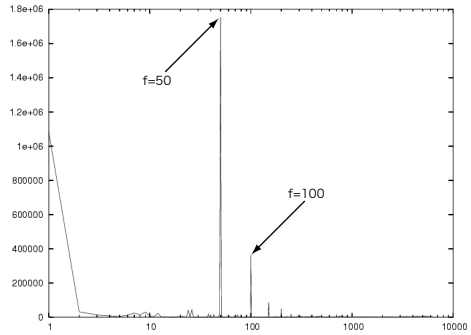

Fig. 1. frequency graph for Sankei Shimbun

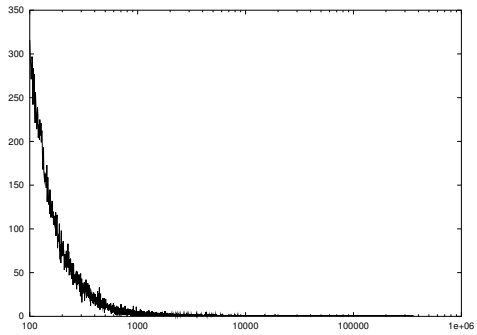

Fig. 2. frequency graph graph for DNA sequences ($f \geq 100$)

is the number of the files. This shows many substrings in the template appear once in all files. But, there is another peak at $f = 100$. This shows some substrings in the template appear twice. Therefore, we can extract substrings which constitute the template of the input files with high probability.

4. EXPERIMENTS

In this section, we show experiments on genome sequences. Compared to HTML files, DNA and amino acid sequences have only little amount of characters and short common substrings. This seems to be undesirable features for substring amplification because short common substrings with few letters do not contain many substrings in them and so they do not constitute sharp peaks in frequency graphs even by substring amplification.

Our purpose of the experiments is to examine the effect of the alphabet size and template width for substring amplification. Therefore, we use both DNA and amino acid sequences in which some constant subsequence with different length are embedded.

The key idea of substring amplification is to use the difference of frequency distributions between template and non-template (background) part. So, we have to use real data for background part when the template part is embedded intensionally. As real data, we used DNA regions of proteins and amino acid subsequences of proteins in the complete genome of Escherichia coli K12 (NC_000913). Therefore, distribution of strings in the background part is natural.

Escherichia coli K12 has 4279 proteins. For each protein, we add some constant sequences and we examine our method can extract the added sequences.

First, we add a constant sequence with length 20 to DNA regions of proteins. Fig. 2 is a frequency graph for these sequences. Although the template with length 20 is embedded in these sequences, we can see no clear peaks because the alphabet size ($|\Sigma| = 4$) is too small and there does not exist many different substrings in the embedded sequence. To see a sharp peak, we need a constant sequence with length about 40.

Next, we add to amino acid sequences. Fig. 3 is a frequency graph for the sequences. In these sequences, the

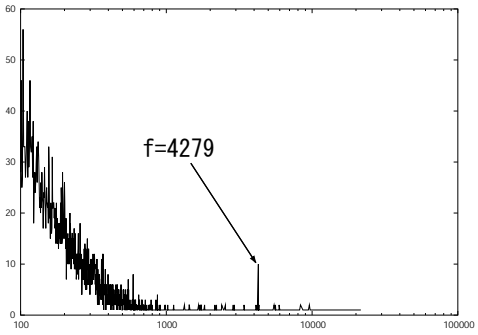

Fig. 3. frequency graph for amino acid sequences ($f \geq 100$)

template with length 8 is embedded. We see that a sharp peek at $f = 4279$, which is the number of proteins. We also see such sharp peaks at the same frequency when the template length is 7 and 6. If the length 5, we can hardly see sharp peaks.

5. CONCLUSION

We examined the applicability of substring amplification, which is an algorithm to the template discovery problem, to genome sequences. Substring amplification utilizes the string frequency distributions. For substring amplification, it is good that there are many different substrings in non-template (background) parts and that input sequences have the long template part. However, genome sequences consist of few characters and seems to have short common substrings.

Our experiments showed that when $|\Sigma| = 4$, a frequency graph does not show sharp peaks even if long template is

embedded. On the other hand, if the alphabet size is large ($|\Sigma| = 20$), we can find a peak when the template width is 6.

Substring amplification can extract a template as a set of substrings which constitute sharp peaks in a frequency graph. So, it is challenging future work to extend the "format" of the pattern so that some ambiguity is allowed, such as `[AC]`.

Counting time for all experiments in Section 4 are about 500 seconds (the data size is about 4.3MB). Currently, our implementation just counts all substrings sequentially. It is another important future work to incorporate some good data structure, such as the suffix array, to the implementation.

6. REFERENCES

[1] Dan Gusfield, *Algorithms on Strings, Trees and Sequence*, Cambridge University Press, New York, 1997.

[2] Makoto Nagao and Shinsuke Mori, "A New Method of N-gram Statistics for Large Number of n and Automatic Extraction of Words and Phrases from Large Text Data of Japanese," in *Proceedings of the 15th International Conference on Computational Linguistics*, 1994, pp. 611–615.

[3] Dana Angluin, "Finding Patterns Common to a Set of Strings," *Journal of Computer and System Sciences*, vol. 21, pp. 46–62, 1980.

[4] Michael Kearns and Leonard Pitt, "A Polynomial-Time Algorithm for Learning k-variable Pattern Languages from Examples," in *Proceedings of the 2nd Annual Workshop on Computational Learning Theory*, 1989, pp. 57–71.

[5] Takeshi Shinohara, "Polynomial Time Inference of Extended Regular Pattern Languages," in *RIMS Symposium on Software Science and Engineering (1982)*. 1983, Lecture Notes in Computer Science 147, pp. 115–127, Splinger-Verlag.

[6] Nicolas Kushmerick, "Wrapper Induction: Efficiency and Expressiveness," *Artificial Intelligence*, vol. 118, pp. 15–68, 2000.

[7] Yasuhiro Yamada, Daisuke Ikeda, and Sachio Hirokawa, "Automatic Wrapper Generation for Multilingual Web resources," in *Proceedings of the 5th International Conference on Discovery Science*. 2002, Lecture Notes in Computer Science 2534, pp. 332–339, Splinger-Verlag.

[8] Rakesh Agrawal and Ramakrishnan Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," in *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, September 1994, pp. 487–499.

[9] Hiroki Arimura and Shinichi Shimozono, "Maximizing Agreement with a Classification by Bounded or Unbounded Number of Associated Words," in *Proceedings of the 9th International Symposium on Algorithms and Computation*. 1998, Lecture Notes in Artificial Intelligence 1533, pp. 39–48, Springer-Verlag.

[10] Tatsuya Asai, Hiroki Arimura, Takeaki Uno, and Shinichi Nakano, "Discovering Frequent Substructures in Large Unordered Trees," in *Proceedings of the 6th International Conference on Discovery Science (to appear)*, 2003.

[11] David Maier, "The Complexity of Some Problems on Subsequences and Supersequences," *Journal of the ACM*, vol. 25, pp. 322–336, 1978.

[12] Daisuke Ikeda, Yasuhiro Yamada, and Sachio Hirokawa, "Pattern Discovery from Distributions of String Frequency (to appear)," in *Proceedings of the 72th IPSJ SIG FI*, September 2003, (in Japanese).

[13] Yasuhiro Yamada, Daisuke Ikeda, and Sachio Hirokawa, "Frequency Analysis of Semi-structured Documents with Structural Similarity," in *Proceedings of the 2nd Forum on Information Technology (FIT2003)*, 2003, pp. 59–60, (in Japanese).

[14] Erika Tateishi, Osamu Maruyama, and Satoru Miyano, "Extracting Best Consensus Motifs from Positive and Negative Examples," in *Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science*. February 1996, Lecture Notes in Computer Science 1046, pp. 219–230, Springer-Verlag.

[15] Hideo Bannai, Shunsuke Inenaga, Ayumi Shinohara, Masayuki Takeda, and Satoru Miyano, "A String Pattern Regression Algorithm and Its Application to Pattern Discovery in Long Introns," in *Genome Informatics (GIW2002)*, 2002, vol. 13, pp. 3–11.