The Probability of Occurrence in a Single Sequence

Ezekiel F. Adebiyi [1]

Keywords: motifs, gene finding, regular expression, deterministic finite automaton(DFA), statistical significance.

1 Introduction

The problem considered here is to determine p_s, the probability that a single random sequence X (whose characters are drawn from the set of characters in Σ), of length L contains at least one occurrence of s that is Edit distance at most D from s. The computation of p_s helps to determine statistical significance in a variety of pattern searches such as motif searches and gene finding[3, 6].

Tompa[6] developed the first direct and efficient algorithm to compute p_s, but for the case of at most one substitution and Markov chains of order 1, with no insertions or deletions. The algorithm takes $O(|X| \cdot |s|^2)$ time. Atteson[3] presented an algorithm for the exact computation of the probability that a random string of a certain length matches a given regular expression, allowing insertions and deletions. The problem considered in [3] is in the class of NP-hard problems, but it is fixed-parameter tractable. Thus his algorithm that runs in time $O(|\Sigma||X|e^{|s|})$, is of linear order in the length of the sequence for small patterns. Note that, the running time is often much smaller, infact polynomial for some restricted forms of regular expressions, than the worst case for most expression which occur in practice. Briefly, Atteson method calculates the probability that a random string of length L, matches a given regular expression by computing the DFA of the regular expression and multiplying the resulting sparse Markov chain transition matrix by the initial vector $|X|$ times.

A possible combination and extension of Atteson[3] and Tompa[6] ideas requires that, we solve a basic problem: the development of regular expressions (henceforth *RE*s) that are able to represent the D-neighborhood of s. For our analysis below, we adopted the D-neighborhood of s according to Edit distance as defined in [5]. Let $\delta(V, W)$ be the Edit distance between V and W. The *D-neighborhood of a string* W is the set of all strings with Edit distance at most D from W, *i.e.,* $N_D(W) = \{V : \delta(V, W) \leq D\}$ and the *condensed D-neighborhood of* W is the set of all strings in the D-neighborhood of W that do not have a prefix in the neighborhood, *i.e.,* $\overline{N_D}(W) = \{V : V \in N_D(W)$ and no prefix of V is in $N_D(W)\}$. The application of the above concept with the combination of [3] and [6] ideas lead to the algorithm outline in the following sections. The remainder of this paper concentrates on the use of Edit distance, but note that the resulting approach is also applicable to Hamming distance. What changes in the whole system is the generation of D-neighborhood of s according to Hamming distance instead of Edit distance.

2 Formulating Regular Expressions and Computation of their single DFA

Given pattern s, the condensed D-neighborhood of s according to the definition above is generated and the words are concatenated but separated by special symbols for recognition, to form a string. We then build a suffix tree for this string. Finally, we use the algorithm of Gusfield[4] for finding left diverse nodes, to group the concatenated words into regular

[1]Department of Computer and Information Technology, College of Science and Technology, Covenant University, P.M.B 1023, Ota, Nigeria. E-mail: `adebiyi@informatik.uni-tuebingen.de`

expression classes, based on some maximal common substring that they shared. No two groups are allow to share the same word, that is, each word belong to only one RE class. Let t be the number of words in each RE class. To avoid over-representation of the D-neighborhood words by their resulting RE classes, let $t \leq |\Sigma|$. Using the regular expression definition of [7](page 238), the resulting REs that represent each class can either be simple or complex. For example, $(A|G|T|C)ATAATA$, $(\epsilon|C|G)ATAATA$ are complex REs, while $CTATAGT$ is a simple RE.

Wilhelm and Maurer[7] in section 7.4.2 presented an algorithm that compute a single DFA for a sequence of regular definitions. To the sequence of regular expression classes derived for a given pattern s, we apply their algorithm to compute the required single DFA.

3 Computation of the p_s and Discussion

Atteson[3] showed how the resulting single DFA above can be use to calculate p_s when the subject sequence is an $i.i.d$ random sequence or a rth-order Markov chain. We adopt his method, as it is over the more general set of all DFA's.

The full application of our method to finding the statistical significance of a pattern, for example, motifs will be done in [1]. We show in tables 1 and 2, how many simple and complex REs can be derived from a given pattern s. This tables also discuss the ratio of the size of condensed neighborhood of s, $\overline{N_D}(s)$, to the number of derivable regular expression classes. In a further task, we dramatically reduced the number of REs derivable from a longer pattern, by considering not all condensed D- neighborhood words of s but those that occur in the input sequences. This has been mentioned earlier on by Tompa[6]. This result is shown in table 2. The patterns used in the following tables are motifs found in at least half of the sequences of B. Subtilis sequences used in [2]. The entries of column 2 of each table indicate the number of complex and simple REs respectively.

Table 1:

s for $D = 1$	# REs	$\|\overline{N_D}(s)\|$
TATAGT	14/1	34
GATATA	12/1	31
GTGACA	15/0	33
GTTGAG	13/1	32
TATAAT	14/0	33

Table 2:

s for $D = 2$	# REs	$r.$ $\|\overline{N_D}(s)\|$
TAAGAAAAA	49/0	124
TATTTAGAA	51/1	124
ATAAATGAA	65/2	158
GTGTTAAAA	48/1	122
CAAATATAA	59/0	132

References

[1] Adebiyi, Ezekiel F., and Kaufmann, M. *Extracting common motifs using consensus and weighted matrix models under the edit distance: Theory and Experimentation.* In preparation, 2004.

[2] Adebiyi Ezekiel F. and Kaufmann, M. *Extracting common motifs under the levenshtein measure: Theory and Experimentation.* 2nd Intl. Workshop, WABI, 140-156, 2002.

[3] Atteson, K. *Calculating the exact probability of language-like patterns in biomolecular sequences.* 6th Intl. Conf. Intelligent Systems for Molecular Biology, 17-24, 1998.

[4] Gusfield D. *Algorithms on strings, trees and sequences.* Cambridge University Press, New York, 1997.

[5] Myers E. *A sub-linear algorithm for approximate keyword matching.* Algorithmica 12, 4-5, 345-374, 1994.

[6] Tompa, M. *An exact method for finding short motifs in sequences, with application to the ribosome binding site problem.* 7th Intl. Conf. Intelligent Systems for Molecular Biology, 262-271, 1999.

[7] Wilhelm, R. and Maurer, D. *Compiler Design.* Addison-Wesley Publishing Company, 1996.