Most Significant Substring Mining Based on Chi-square Measure

Sourav Dutta and Arnab Bhattacharya

Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur, India
{sdutta,arnabb}@iitk.ac.in

Abstract. Given the vast reservoirs of sequence data stored worldwide, efficient mining of string databases such as intrusion detection systems, player statistics, texts, proteins, etc. has emerged as a great challenge. Searching for an unusual pattern within long strings of data has emerged as a requirement for diverse applications. Given a string, the problem then is to identify the substrings that differs the most from the expected or normal behavior, i.e., the substrings that are statistically significant (i.e., less likely to occur due to chance alone). To this end, we use the chi-square measure and propose two heuristics for retrieving the top-k substrings with the largest chi-square measure. We show that the algorithms outperform other competing algorithms in the runtime, while maintaining a high approximation ratio of more than 0.96.

1 Motivation

A recent attractive area of research has been detecting statistically relevant sequences or mining interesting patterns from a given string [1,2]. Given an input string composed of symbols from an alphabet set with a probability distribution defining the chance of occurrence of the symbols we would like to find the portions of the string which deviate from the expected behavior and can thus be potent sources of study for hidden pattern and information. An automated monitoring system such as a cluster of sensors sensing the temperature of the surrounding environment for fire alert, or a connection server sniffing the network for possible intrusion detection provides a few of the applications where such pattern detection is essential. Other applications involve text analysis of e-mails and blogs to predict terrorist activities or judging prevalent public sentiments and studying trends of the stock market. Similarly, another interesting field of application can be the identification of good and bad career patches of a sports icon. For example, given the runs scored by Sachin Tendulkar in each innings of his one-day international cricket career, we may be interested in finding his in-form and off-form patches.

A statistical model is used to determine the relationship of an experimental or observed outcome with the factors affecting the system, or to establish the occurrence as pure chance. An observation is said to be statistically significant

M.J. Zaki et al. (Eds.): PAKDD 2010, Part I, LNAI 6118, pp. 319–327, 2010.

if its presence cannot be attributed to randomness alone. The degree of uniqueness of a pattern can be captured by several measures including the p-value and z-score [3,4]. For evaluating the significance of a substring, it has been shown that the p-value provides a more precise conclusion as compared to that by the z-score [1]. However, computing the p-value entails enumerating all the possible outcomes, which can be exponential in number, thus rendering it impractical. So, heuristics based on branch-and-bound techniques have been proposed [5]. The *log–likelihood ratio* G^2 provides such a measure based on the extent of deviation of the substring from its expected nature [6]. For multinomial models, the χ^2 statistic approximates the importance of a string more closely than the G^2 statistic [6,7]. Existing systems for intrusion detection use multivariate process control techniques such as Hotelling's T^2 measure [8], which is again computationally intensive. The chi-square measure, on the other hand, provides an easy way to closely approximate the p-value of a sequence [6]. To simplify computations, the χ^2 measure, unlike Hotelling's method, does not consider multiple variable relationship, but is as effective in identifying "abnormal" patterns [2]. Thus, in this paper, we use the Pearson's χ^2 statistic as a measure of the p-value of a substring [6,7]. The χ^2 distribution is characterized by the *degrees of freedom*, which in the case of a string, is the size of the alphabet set minus one. The larger the χ^2 value of a string, the smaller is its p-value, and hence more is its deviation from the expected behavior.

Formally, given a string S composed of symbols from the alphabet set Σ with a given probability distribution P modeling the chance of occurrence of each symbol, the problem is to identify and extract the top-k substrings having the maximum chi-square value or the largest deviation within the framework of p-value measure for the given probability distribution of the symbols. Naïvely we can compute the χ^2 value of all the substrings present in S and determine the top-k substrings in $O(l^2)$ time for a string of length l. We propose to extract such substrings more efficiently.

Related Work: The blocking algorithm and its heap variant [9] reduce the practical running time for finding such statistically important substrings, but suffers from a high worst-case running time. The number of blocks found by this strategy increases with the size of the alphabet set and also when the probabilities of the occurrence of the symbols are nearly similar. In such scenarios, the number of blocks formed can be almost equal to the length of the given string, thereby degenerating the algorithm to that of the naïve one. The heap variant requires a high storage space for maintaining the separate *max* and *min* heap structures and also manipulates a large number of pointers. Further, the algorithm cannot handle top-k queries. In time-series databases, categorizing a pattern as surprising based on its frequency of occurrence and mining it efficiently using suffix trees has been proposed in [10]. However, the χ^2 measure, as discussed earlier, seems to provides a better parameter for judging whether a pattern is indeed interesting.

In this paper, we propose two algorithms, *All-Pair Refined Local Maxima Search* (ARLM) and *Approximate Greedy Maximum Maxima Search* (AGMM)

to efficiently search and identify interesting patterns within a string. We show that the running time of the algorithms are better than the existing algorithms with lesser space requirements. The procedures can also be easily extended to work in streaming environments. ARLM, a quadratic algorithm in the number of local maxima found in the input string, and AGMM, a linear time algorithm, both use the presence of local maxima in the string. We show that the approximation ratio of the reported results to the optimal is 0.96 or more. Empirical results emphasize that the algorithms work efficiently.

The outline of the paper is as follows: Section 2 formulates the properties and behavior of strings under the χ^2 measure. Section 3 describes the two proposed algorithms. Section 4 discusses the experimental results, before Section 5 concludes the paper.

2 Definition and Properties

Let $S = s_1 s_2 \dots s_l$ be a given string of length l composed of symbols s_i from the alphabet set $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_m\}$, where $|\Sigma| = m$. To each symbol $\sigma_i \in \Sigma$ is associated a p_{σ_i} (henceforth represented as p_i), denoting the probability of occurrence of that symbol, such that $\sum_{i=1}^{m} p_i = 1$. Let $\theta_{\sigma_i,S}$ (henceforth represented as $\theta_{i,S}$) denote the observed number of symbol σ_i in string S.

The chi-square value of a string $S \in \Sigma^*$ of length l is computed as

$$\chi^2_S = \sum_{i=1}^{m} \frac{(p_i l - \theta_{i,S})^2}{p_i l} \tag{1}$$

We now state certain observations and lemmas, the formal proofs of which are in the full version of the paper [11].

Observation 1. *Under string concatenation operation (.), for two arbitrary strings S and T drawn from the same alphabet set and probability distribution of the symbols (henceforth referred to as the* same universe*), the χ^2 measure of the concatenated string is commutative, i.e., $\chi^2_{S.T} = \chi^2_{T.S}$.*

Lemma 1. *The χ^2 value of the concatenation of two strings drawn from the same universe is less than or equal to the sum of the χ^2 values of the individual strings.*

Lemma 2. *The χ^2 value of a string composed of only a single type of symbol increases with the length of the string.*

We next define the term *local maxima* and describe the procedure for finding such a local maxima within a given string.

Definition 1 (Local maxima). *The* local maxima *is a substring, such that while traversing through it, the inclusion of the next symbol does not decrease the χ^2 value of the resultant sequence.*

Let $s_1 s_2 \dots s_n$ be a local maxima of length n. Then the following holds: $\chi^2_{s_1 s_2} \geq \chi^2_{s_1}, \dots, \chi^2_{s_1 s_2 \dots s_n} \geq \chi^2_{s_1 s_2 \dots s_{n-1}}$ and $\chi^2_{s_1 s_2 \dots s_{n+1}} \leq \chi^2_{s_1 s_2 \dots s_n}$.

The process of finding the local maxima involves a single scan of the entire string. We consider the first local maxima to start at the beginning of the string. We keep appending the next symbol to the current substring until there is a decrease in the chi-square value of the new substring. The present substring is then considered to be a local maxima ending at the previous position. The last symbol appended that decreased the chi-square value signifies the start of the next local maxima. Thus, the running time is $O(l)$ time for a string of length l.

Lemma 3. *The expected number of local maxima in a string of length l is $O(l)$.*

However, practically the number of local maxima is less than l, as all adjacent positions of dissimilar symbols may not correspond to a local maxima boundary. We further optimize the local maxima finding procedure by initially *blocking* the string S, as described in [9], and then searching for the local maxima. A contiguous sequence of the same symbol is considered to be a block, and is replaced by a single instance of that symbol representing the block. If a symbol is selected, the entire block associated with it is considered to be selected. The next lemma shows that a local maxima cannot end in the middle of a block.

Lemma 4. *If the insertion of a symbol of a block increases the chi-squared value of the current substring, then the chi-squared value will be maximized if the entire block is inserted.*

3 Algorithms

Based on the observations, lemmas and local maxima extracting procedure discussed previously, in this section we explain the *All-Pair Refined Local Maxima* (ARLM) and *Approximate Greedy Maximum Maxima* (AGMM) search algorithms for mining the most significant substring based on the chi-square value.

3.1 All-Pair Refined Local Maxima Search (ARLM)

Given a string S of length l and composed of symbols from the alphabet set Σ, we first extract all the local maxima present in it in linear time, as described earlier. With S partitioned into its local maxima, the global maxima can either start from the beginning of a local maxima or from a position within it. Thus, it can contain an entire local maxima, a suffix of it or itself be a substring of a local maxima. It is thus intuitive that the global maxima should begin at a position such that the subsequent sequence of characters offer the maximum chi-square value. Otherwise, we could keep adding to or deleting symbols from the front of such a substring and will still be able to increase its χ^2 value. Based on this, the ARLM heuristic finds within each local maxima the suffix having the maximum

chi-square value, and considers the position of the suffix as a potential starting point for the global maxima.

Let xyz be a local maxima, where x is a prefix, y is a single symbol at position ψ, and z is the remaining suffix. For a local maxima the chi-square value of all its suffices is computed. The starting position of the suffix having the maximum chi-square value provides the position ψ for the component y, i.e, yz will be the suffix of xyz having the maximum chi-square value.

For each local maxima, the position ψ is inserted into a list α. If no such proper suffix exists for the local maxima, the starting position of the local maxima xyz is inserted in the list. After populating α with position entries of y for each of the local maxima of the input string, the list contains the prospective positions from where the global maxima may start.

The string S is now reversed and the same algorithm is re-run. This time, the β list is similarly filled with positions y' relative to the beginning of the string.

For simplicity and efficiency of operations, we maintain a table C having m rows and l columns, where m is the cardinality of the alphabet set. The rows of the table contain the observed number of each associated symbols present in the length of the string denoted by the column. The observed count of a symbol between two given positions of the string can be easily computed from this table in $O(1)$ time.

Given the two α and β lists, we now find the chi-square value of substrings from position $g \in \alpha$ to $h \in \beta$, and $g \leq h$. The substring having the maximum value is reported as the global maxima. While computing the chi-square values for all the pairs of positions in the two list, the top-k substrings can be maintained using a heap of k elements.

Conjecture 1. The starting position of the global maxima is always present in the α list.

Corollary 1. *From the above conjecture, it follows that the ending position of the global maxima is also present in the β list.*

Finding all the local maxima in the string requires a single pass, which takes $O(l)$ time for a string of length l. Let the number of local maxima in the string be d. Finding the maximum valued suffix for each local maxima using the C table, requires another pass of each of the local maxima, and thus also takes $O(l)$ time. Since, each local maximum contributes one position to the lists, the number of elements in both the lists is d. We then evaluate the substrings formed by each possible pair of start and end positions, which takes $O(d^2)$. So, in total, the time complexity of the algorithm is $O(l + d^2)$.

3.2 Approximate Greedy Maximum Maxima Search (AGMM)

In this section, we propose a linear time greedy algorithm for finding the maximum substring, which is linear in the size of the input string S. We extract all the local maxima of the input string and its reverse, and populate the α and

β lists as discussed previously. We identify the local maxima suffix max having the maximum chi-square value among all the local maxima present in the string. AGMM assumes this local maxima suffix to be completely present within the global maxima. We then find a position $g \in \alpha$ for which the new substring starting at g and ending with max as a suffix has the maximum χ^2 value, for all g. Using this reconstructed substring, we find a position $h \in \beta$ such that the new string starting at the selected position g and ending at position h has the maximum chi-square measure for all positions of h. This new substring is reported by the algorithm as the global maxima.

The intuition here is that the global maxima will contain the maximum of the local maxima to maximize its value. Although this is a heuristic, the assumption is justified by empirical results in Section 4, which shows that we always obtain an approximation ratio of 0.96 or more.

Using the C table, AGMM takes $O(d)$ time, where d is the number of local maxima found. The total running time of the algorithm is thus $O(d + l)$. Thus, being a linear time algorithm, it provides a order of increase in the runtime as compared to the other algorithms.

4 Experiments

To assess the performance of the two proposed heuristics ARLM and AGMM, we conduct tests on multiple datasets and compare it with the results of the naïve algorithm and the blocking algorithm [9]. The heap variant of the blocking algorithm is not efficient as it has a higher running time and uses more memory, and hence has not been reported. We compare the results based on the following parameters: (i) search time for top-k queries, (ii) number of local maxima found, and (iii) accuracy of the result based on the ratio of the optimal χ^2 value to that returned by the algorithms.[1]

Table 1. Results of (a) Sachin's records. (b) Uniform dataset

Form	Date	Avg.	Runs scored
Best patch	22/04/98 to 13/11/98	84.31	143,134,33,18,100* 65,53,17,128,77 127*,29,2,141,8,3 118*,18,11,124*
Worst patch	15/03/92 to 19/12/92	21.89	14,39,15 10,22,21 32,23,21

(a)

Parameters	Variable	Blocks	Local maxima
m=5, k=1	l=10^3	831	742
	l=10^4	7821	6740
	l=10^5	77869	66771
l=10^4, k=1	m=5	7821	6740
	m=25	8104	7203
	m=50	8704	7993

(b)

[1] The experiments were conducted on a 2.1GHz desktop PC with 2GB of memory using C++ in Linux.

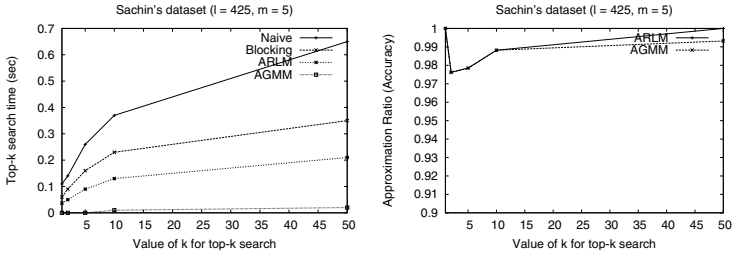

Fig. 1. (a) Time for finding the top-k query in Sachin's run dataset. (b) Approximation ratio of the top-k query in Sachin's run dataset.

4.1 Real Datasets

We used the innings-by-innings runs scored by Sachin Tendulkar in one-day internationals (ODI)[2] as a real dataset. We quantized the runs scored into 5 symbols as follows: 0-9 is represented by A, 10-24 by B, 25-49 by C, 50-99 by D, and 100+ by E. The actual probability of occurrences of the different symbols were 0.28, 0.18, 0.22, 0.22 and 0.10 respectively for the five symbols, and the overall average is 45.5 runs per innings. Table 1(a) summarizes the results. We find that during his best patch he had scored 8 centuries and 3 half-centuries in 20 innings with an average of 84.31, while in the off-form period he had an average of 21.89 in 9 innings without any score of above 40.

Figure 1(a) show the times taken by the different algorithms for different values of k. The AGMM algorithm requires the least running time as compared to the other procedures, while the ARLM is faster than the naïve and the blocking ones. The number of local maxima found was 281, which is lesser than 319, the number of blocks constructed by the blocking algorithm. So, the heuristic prunes the search space more efficiently. Figure 1(b) plots the approximation factor for the heuristics. The accuracy of the ARLM heuristic is found to be 100% for the top-1 query, i.e., it provides the correct result validating the conjecture we proposed in Section 3. As the value of k increases we find an increase in the approximation ratio of both the heuristics.

4.2 Synthetic Datasets

We now benchmark the ARLM and AGMM heuristics against datasets generated randomly using a uniform distribution. To simulate the deviations from the expected characteristics as observed in real applications, we perturb the random data with chunks of data generated from a geometric distribution with parameter $p = 0.3$. The parameters that affect the performance of the heuristics are: (i) length of the input string, l, (ii) size of the alphabet set, m, and (iii) number of top-k values. For different values of these parameters we compare our

[2] http://stats.cricinfo.com/ci/engine/player/35320.html?class=2;template=results;type=batting;view=innings

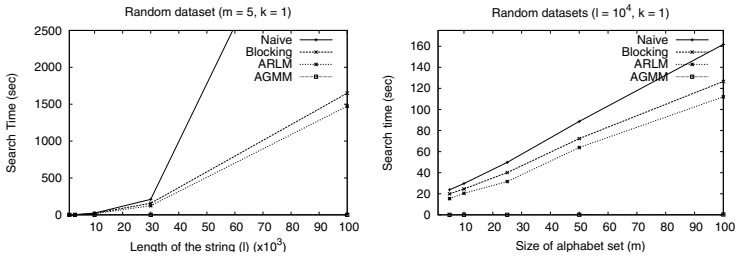

Fig. 2. (a) Effect of length on search time. (b) Effect of size of alphabet on search time.

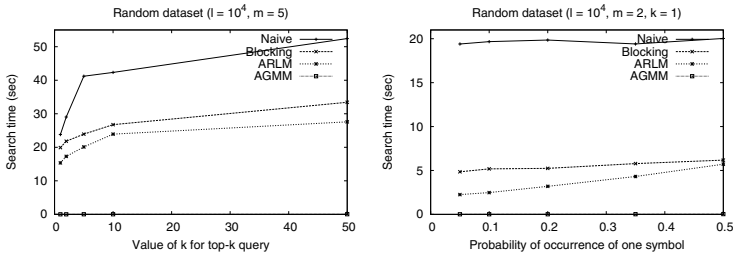

Fig. 3. (a) Effect of value of k for top-k query on search time. (b) Effect of probability in two symbol string on search time.

algorithms with the existing ones on the basis of (a) time to search, (b) approximation ratio of the results, and (c) the number of blocks evaluated in case of blocking algorithm to the number of local maxima found by our algorithm.

Fig 2(a) shows that with the increase in the length of the input string l, the time taken for searching the top-k queries increases. The number of blocks or local maxima increases with the size of the string and, hence, the time increases. The time increases more or less quadratically for ARLM and the other existing algorithms according to the analysis shown in Section 3.1. ARLM takes less running time than the other techniques, as the number of local maxima found is less than the number of blocks found by the blocking algorithm (see Table 1(b)). Hence, it provides better pruning of the search space and is faster. On the other hand, AGMM being a linear time heuristic, runs an order of time faster than the others. We also find that the accuracy of the top-k results reported by AGMM show an improvement with the increase in the string length (graph not shown). The approximation factor for ARLM is 1 for the top-1 query in all the cases tested, while for other top-k queries and for AGMM it is always above 0.96.

The time taken for searching the top-k query as well as the number of blocks formed increases with the size of the alphabet m (Table 1(b) and Fig 2(b)). As m increases, the number of blocks increases as the probability of the same symbol occurring contiguously falls off. A local maxima can only end at positions containing adjacent dissimilar symbols. So the number of local maxima found

increases as well. There seems to be no appreciable effect of m on the approximation ratio of the results returned by the algorithms. We tested with varying values of m with $l = 10^4$ and $k = 2$, and found the ratio to be 1 in all cases.

Fig 3(b) shows the effect of varying probability of occurrence of one of the symbols in a string composed of two symbols only. The approximation ratio remained 1 for both heuristics for the top-1 query.

We next show the scalability of our algorithms by conducting experiments for varying values of k for top-k substrings. Fig 3(a) shows that search time increases with the increase in the value of k. This is expected as more computations are performed. The accuracy of the results for the heuristics increases with k. For $k = 2$, it is 0.96, and increases up to 1 when k becomes more than 10.

5 Conclusions

In this paper, we proposed the problem of finding top-k substrings within a string with the maximum chi-square value for mining interesting patterns. The chi-square value represents the deviation of the observed from the expected. We used the concept of local maxima and proposed two efficient heuristics that run in time quadratic and linear in the number of local maxima. Experiments showed that the heuristics are faster than the existing algorithms, are scalable, and return results that have an approximation ratio of more than 0.96.

References

1. Denise, A., Regnier, M., Vandenbogaert, M.: Accessing the statistical significance of overrepresented oligonucleotides. In: Work. Alg. Bioinf. (WABI), pp. 85–97 (2001)
2. Ye, N., Chen, Q.: An anomaly detection technique based on chi-square statistics for detecting intrusions into information systems. Quality and Reliability Engineering International 17(2), 105–112 (2001)
3. Rahmann, S.: Dynamic programming algorithms for two statistical problems in computational biology. In: Work. Alg. Bioinf. (WABI), pp. 151–164 (2003)
4. Regnier, M., Vandenbogaert, M.: Comparison of statistical significance criteria. J. Bioinformatics and Computational Biology 4(2), 537–551 (2006)
5. Bejerano, G., Friedman, N., Tishby, N.: Efficient exact p-value computation for small sample, sparse and surprisingly categorical data. J. Comp. Bio. 11(5), 867–886 (2004)
6. Read, T., Cressie, N.: Goodness-of-fit statistics for discrete multivariate data. Springer, Heidelberg (1988)
7. Read, T., Cressie, N.: Pearson's χ^2 and the likelihood ratio statistic G^2: a comparative review. International Statistical Review 57(1), 19–43 (1989)
8. Hotelling, H.: Multivariate quality control. Techniques of Statistical Analysis 54, 111–184 (1947)
9. Agarwal, S.: On finding the most statistically significant substring using the chi-square measure. Master's thesis, Indian Institute of Technology, Kanpur (2009)
10. Keogh, E., Lonardi, S., Chiu, B.: Finding surprising patterns in a time series database in linear time and space. In: SIGKDD, pp. 550–556 (2002)
11. Dutta, S., Bhattacharya, A.: Mining most significant substrings based on the chi-square measure. arXiv:1002.4315 [cs.DB] (2010)