New Generation Computing, 18(2000)49-60
Ohmsha, Ltd. and Springer-Verlag

Efficient Discovery of Optimal Word-Association Patterns in Large Text Databases

Shinichi SHIMOZONO
Dept. of Artificial Intelligence, Kyushu Inst. of Tech., Iizuka 820–8502, Japan
sin@ai.kyutech.ac.jp

Hiroki ARIMURA
Dept. of Informatics, Kyushu Univ., Fukuoka 812-8581, Japan
Precursory Research for Embryonic Science and Technology at Japan Science and Technology Corporation
arim@i.kyushu-u.ac.jp

Setsuo ARIKAWA
Dept. of Informatics, Kyushu Univ., Fukuoka 812-8581, Japan
arikawa@i.kyushu-u.ac.jp

Received 03 September 1999

Abstract We study efficient discovery of proximity word-association patterns, defined by a sequence of strings and a proximity gap, from a collection of texts with the positive and the negative labels. We present an algorithm that finds all d-strings k-proximity word-association patterns that maximize the number of texts whose matching agree with their labels. It runs in expected time complexity $O(k^{d-1}n\log^d n)$ and space $O(k^{d-1}n)$ with the total length n of texts, if texts are uniformly random strings. We also show that the problem to find one of the best word-association patterns with arbitrarily many strings is MAX SNP-hard.

Keywords: Text Databases, Data Mining, Optimization, Proximity Word-association Patterns, Discovery Science.

§1 Introduction

In this paper, we present a fast algorithm for efficient discovery of combinatorial patterns, called *proximity word-association patterns*, from a collection of texts with binary classification labels.

Since emerged in early 1990's, data mining has been extensively studied to develop semi-automatic tools for discovering valuable rules from stored facts

in large scale databases.[1] A rule which is looked for is an association among attributes that gives a useful property. Mainly a considerable amount of results have been known for well-defined and structured databases, such as relational databases with boolean or numeric attributes.[1,7]

Beside this, recent progress of measuring and sensing technology, storage devises and network infrastructure has been rapidly increasing the size and the species of weakly-structured databases such as bibliographic databases, e-mails and HTML streams and raw experimental results as genomic sequences. These lead to potential demands to data mining tools for databases where no attributes or structure is assumed in advance. However, there are still a few results in this direction.[14] One difficulty might be that a tool should quickly extract a structure behind the data as well as discover rules of interests. Our aim is to develop an efficient tool for text databases along the lines of data mining.

We consider a data mining problem in a large collection of unstructured texts based on association rules over subwords of texts. A *proximity word association pattern* is an expression such as (TATA, TAGT, AGGAGGT; 30) that expresses a rule that if subwords TATA, TAGT, and AGGAGGT appear in a text in this order with distance no more than 30 letters then a specified property ξ will hold over the text with a probability. Note that we use the term *word* in the sense of a consecutive substring of unlimited length rather than a *keywords*.

The data mining problem we consider is the *maximum agreement problem* defined as follows. Assume that we are given a collection of documents with an *objective condition*, that is, a binary label ξ over texts in S that indicates if a text has a property of interest. A pattern π agrees with ξ on s if π matches s if and only if $\xi(s) = 1$. The maximum agreement problem, also called *empirical risk minimization*,[5] is to find a k-proximity d-word association pattern π that maximizes the number of documents in S on which π agrees with ξ. This problem is an instance of *optimal pattern discovery*, which attracts much attention in the field of data mining.[7]

The notion of proximity word association patterns extends frequently used *proximity patterns* consisting of two strings and a gap.[10] An algorithm that efficiently solves this problem can be applied in a wide range of practical problems and plays a key role in the discovery of a consensus motif from protein sequences as in Reference.[14] Further, the maximum agreement problem plays an important role in computational learning theory; it is shown that an algorithm that efficiently solves the problem for a class with moderate complexity will be an efficient learner with the same class in the framework of *Agnostic PAC-learning*.[9]

Clearly, the maximum agreement problem by proximity word-association patterns is polynomial-time solvable in $O(n^{2d+1})$ time if the patterns are formed from at most d strings. A modified algorithm that uses the suffix tree structure improves to $O(n^{d+1})$ time but still requires $O(n^d)$ scans of input texts.[14] However, the practical importance of the problem requires more efficient, essentially faster algorithm. Hence, we have devised an algorithm that efficiently solves the maximum agreement problem, which finds a d-words k-proximity word-association pattern with the maximum agreement in expected running time $O(k^{d-1}n\log^{d+1}n)$ and space $O(k^{d-1}n)$ with the total length n of texts,

if texts are uniformly random strings. Even in the worst case the algorithm runs in time $O(k^d n^{d+1} \log n)$ which is essentially faster than the naive method.

On the other hand, if the number of strings in a proximity word-association pattern is not limited, the maximum agreement problem is intractable. Dealing with this hardness, we should look for a polynomial-time algorithm that solves the problem approximately with some guaranteed approximation ratio. We partially clarify this issue by presenting the nonapproximability of our maximization problem. We show that the maximum agreement problem of word-association patterns formed from arbitrary many words is hard to approximate within a factor arbitrarily close to one in polynomial time, i.e., has no polynomial-time approximation scheme (PTAS) unless P=NP. It is an interesting contrast that the problem is obviously approximable in polynomial-time within a fixed approximation ratio.

The remainder of this paper is organized as follows. First, we introduce some notions and definitions. Next, we present the efficient algorithm that discovers all the best word-association patterns, and analyze its running time. Then we prove the nonapproximability of the problem. We end with some discussion on the relation between our problem and computational learning theory.

§2 Notions and Definitions

The set Σ is a finite alphabet throughout this paper. We assume some fixed total ordering on the letters in Σ. The empty string in Σ^* whose length is zero is referred to ϵ. For a string $s \in \Sigma^*$, we denote by $|s|$ the length of s and by $s[i]$ with $1 \leq i \leq |s|$ the ith letter of s. The concatenation of two strings s and t in Σ^* is denoted by $s \cdot t$, or simply by st. If for a string t there exist (possibly empty) strings $u, v \in \Sigma^*$ such that $t = u \cdot v$, then we say that u and v are a *prefix* and a *suffix* of t, respectively. Any prefix of a suffix of t is a *subword* (*substring*) of t. A notion $t[i, j]$ refers to the subword $t[i] \cdots t[j]$ of t from the ith to the jth letters. An *occurrence of a string v in t* is an integer $1 \leq i \leq |s|$ specifying a position from which the substring v occurs in t, that is, an index i such that $t[i, i + |v| - 1] = v$.

A *proximity word-association pattern π over Σ* (*word-association pattern*) is a pair $\pi = \langle (w_1, \ldots, w_d), k \rangle$ of a finite sequence of strings in Σ^* and a nonnegative integer k called *proximity*. A *(d, k)-pattern* refers to a d-word k-proximity association pattern. An occurrence of π in s is a sequence $(i_1, \ldots, i_d)$ of occurrences of $w_1, \ldots, w_d$ in s that satisfy $0 < i_{j+1} - i_j \leq k$ for all $1 \leq j < d$. We say *π matches* $s \in \Sigma^*$ if an occurrence of π in s exists. When we specify the positive infinity proximity ∞, we write $\pi = (w_1, \ldots, w_d)$ by omitting k. The concatenation $\pi \cdot \tau$ of two patterns $\pi = \langle (u_1, \ldots, u_c), k \rangle$ and $\tau = \langle (w_1, \ldots, w_d), k \rangle$ with the same proximity is the pattern $\pi \cdot \tau = \langle (u_1, \ldots, u_c, w_1, \ldots, w_d), k \rangle$.

Now we define our problem. A *sample* is a finite set $S = \{s_1, \ldots, s_m\}$ of strings in Σ^*, and an *objective condition over S* is a binary labeling function $\xi : S \to \{0, 1\}$. A string s_i in S is a *document* whose label is $\xi(s_i)$. We say a word-association pattern π *agrees with ξ on s_i* if either (i) π matches s_i and $\xi(s_i) = 1$, or (ii) π does not match s_i and $\xi(s_i) = 0$. The number of elements in S is denoted by $|S|$.

Definition 2.1
MAX AGREEMENT BY k-PROXIMITY d-WORD ASSOCIATION PATTERN
An instance (S, ξ) is a pair of a sample $S \subseteq \Sigma^*$ and an objective condition ξ over S. A solution is a d-word k-proximity pattern π over Σ, and the measure of π is the number of documents in S on which π agrees with ξ. The goal of the problem is to find a (d, k)-pattern π of the maximum measure.

The problem for which the proximity and the number of words are not specified is denoted by MAX AGREEMENT BY PROXIMITY WORD ASSOCIATION PATTERN.

Before describing the algorithm, let us briefly review a key data structure, a *suffix tree*.[11] Let t be a string in Σ^* of length n, and let $t[p, *]$ for $1 \leq p \leq n$ be the suffix of t starting at the position p. Then the *suffix tree Tree*(t) *for* t is a rooted tree that satisfies the following: (i) Each edge is labeled by a subword w of t. All edges from the same node are mutually distinguished and alphabetically sorted by the first letters of the labels. (ii) Each node v represents a subword *word*(v) of t defined by the concatenation of the labels on the path from the root to v. (iii) All the non-empty suffices of t are represented by the leaves of *Tree*(S), and each leaf represents the ith suffix $t[i, *]$ of t for some $1 \leq i \leq n$ uniquely. (iv) Every internal node has at least two children.

The suffix tree *Tree*(t) has exactly n leaves and at most $n - 1$ internal nodes, and can be represented in $O(n)$ space. McCreight[11] gives an elegant algorithm that computes *Tree*(t) in linear time and space. The *height* of *Tree*(t) is the length of a longest path from the root to a leaf. The expected height of suffix trees is known to be logarithmistic for a random string.[5]

We employ the suffix tree *Tree*(t) as a compact representation of dictionary of subwords of t with the following notions. Let s be a subword of t. Then the *locus loc*(s) *of* s in *Tree*(t) is the node v in *Tree*(t) such that (i) s is equal to or a prefix of *word*(v), and (ii) for the parent w of v *word*(w) represents a strictly shorter subword of s. Let l_i be the ith leave of *Tree*(t) and let p_i be the position in t such that *word*$(l_i) = t[p_i, *]$. Then $t[p_i, *]$ is the suffix of rank i in the lexicographic order. We refer this starting position p_i to *spos*(i) through the table *spos* of n positive integers, and define *lex* to be its inverse table, that is, *lex*(i) for an index $1 \leq i \leq n$ gives the lexicographic rank of the suffix from the position i. These are called the *suffix arrays*.[10]

In the following discussion, a suffix tree of a string and its notions are extended to those of a set of strings. For this sake, we introduce a special delimiter symbol \$ which satisfies $\$ \notin \Sigma$ and $\$ \neq \$$. Given a sample S and a labeling function ξ, the algorithm builds a suffix tree *Tree*(S) for all documents in S by assuming every documents ending with additional \$. Any edge label including \$ must be on a lowest branching edge. We implicitly regard a sample $S = \{s_1, \ldots, s_m\}$ as a single but separated text $\hat{S} = s_1\$s_2\$\cdots\$s_m\$$ concatenating all the documents. A position $1 \leq p \leq |\hat{S}|$ on the jth document in $\hat{S}$ is identified with the pair (i, j) of the corresponding position i in s_j appropriately.

A pattern π is *canonical* if it can be represented by some sequence of nodes $u_1, \ldots, u_d$ of *Tree*(S) as $\pi = \langle (word(u_1) \ldots, word(u_d)), k \rangle$. Here, if u_j is

a leaf then we define $word(u_j)$ to be the whole string represented by u_j except the delimiter \$. Let (S, ξ) be an instance of our maximum agreement problem, and let π be a pattern. We denote by $count_\alpha(\pi)$ for $\alpha \in \{0, 1\}$ the number of documents that match π and are labeled with α. Then the maximization of the agreement equals the maximization of the difference $\Delta_{S,\xi}(\pi) = count_1(\pi) - count_0(\pi)$ since the number of 0-labeled documents in S is fixed. Let $\omega_S \in \Sigma^*$ be an arbitrary string which is strictly longer than any document in S. Clearly $\Delta_{S,\xi}(\langle(\omega_S), k\rangle)$ equals 0.

The following lemma tells us that it is sufficient to consider only the canonical patterns or ω_S for solving our problem.

Lemma 2.1
Either a canonical pattern or ω_S maximizes $\Delta_{S,\xi}$ over all (d, k)-patterns.

Proof
Let w be a string in Σ^*. By a contradiction, it can be seen that every occurrence of w in S is also an occurrence of $word(loc(w))$. The value $\Delta_{S,\xi}(\pi)$ is calculated with the set of occurrences of π in S and ξ, and each occurrence is a sequence of occurrences of the d-words in π. So we can examine with canonical patterns and ω_S all the measures of (i) patterns that match some documents and (ii) a pattern that match no documents. ■

§3 Computing the Maximum Agreement by Association of a Bounded Number of Words

As well as the fact shown in the previous section, our algorithm employs the following observations: (i) in reality, the height of a suffix tree grows rather slowly with the size of the sample, especially logarithmistically for uniformly random strings; (ii) an internal node of a suffix tree and the leaves descended from it are corresponding to a subword in a document. The algorithm is outlined in Fig. 1. It searches for and tests canonical patterns as hyper-rectangles on d-dimensional space, where the labeled points corresponding to the documents are distributed. This realizes an average case efficient algorithm for random inputs with the fact that we have to visit only a small fraction $O(k^{d-1}nh^d)$ of patterns from $O(n^d)$ possible combinations of d substrings of S, where h is the height of the suffix tree for the input text.

For nonnegative integers i, j with $i \leq j$, we denote by $[i, j]$ the interval from i to j. We define $[i, j]$ to be empty if $i > j$. Then $[i_1, j_1] \times \cdots \times [i_d, j_d]$ with integers $i_1, j_1, \ldots, i_d, j_d$ defines a *d-dimensional axis-parallel rectangle*, or *d-dimensional box*. For a sample S of m documents, the *d-dimensional diagonal set* $Diag_{d,k}(S) \subseteq [1, n]^d \times [1, m]$ *of width* k is the set of labeled points such that a point $(p_1, \ldots, p_d; i)$ in $Diag_{d,k}(S)$ is formed from a sequence $(p_1, \ldots, p_d)$ which is possibly an occurrence of some (d, k)-pattern in s_i. In another words, $Diag_{d,k}(S)$ is the set of all the points $(p_1, \ldots, p_d; i) \in [1, n]^d \times [1, m]$ such that $1 \leq p_j \leq |s_i|$ for $1 \leq j \leq d$ and $0 < p_{j+1} - p_j \leq k$ with $1 \leq j < d$ for $1 \leq i \leq m$. Then *d-dimensional rank space* $Rank_{d,k}(S) \subseteq [1, n]^d \times [1, m]$ *of* S is defined by

$$Rank_{d,k}(S) = \left\{(lex(p_1), \ldots, lex(p_d); i) \mid (p_1, \ldots, p_d; i) \in Diag_{d,k}(S)\right\} .$$

Algorithm *Maximum_Agreement*$_{d,k}$:
Input: A sample $S \subseteq \Sigma^*$ and an objective condition $\xi : S \to \{0,1\}$.
Output: A (d,k)-proximity pattern π over Σ that maximizes $\Delta_{S,\xi}(\pi)$.

1. Compute the suffix tree *Tree*(S), suffix arrays *lex* and *spos* $= lex^{-1}$.
2. Generate the elements of *Diag*$_{d,k}(S)$, then collect them in *Rank*$_{d,k}(S)$.
3. Sort the points in *Rank*$_{d,k}(S)$ lexicographically into the list Q.
4. Compute the levels of all the nodes of *Tree*(S), and initialize the global d Boolean flags for all the nodes of *Tree*(S).
5. Call *Discover*(*Tree*$(S), Q, 1, \pi_0$) with the empty pattern $\pi_0 := \langle(), k\rangle$.
6. Output the list the best patterns π found if the maximum of Δ is positive. Otherwise, return ω_S.

Fig. 1 The Algorithm for Computing the Maximum Agreement over (d,k)-patterns

Let v be any node of *Tree*(S), and let l_1 and l_2 be the left-most and right-most leaves among the descendants of v. We identify a leaf of the suffix tree *Tree*(S) with its occurrence in S. Then we can see that the set of the occurrences of *word*(v) form a consecutive subinterval $I(v) = [lex(l_1), lex(l_2)]$ of $[1,n]$. We associate each canonical (d,k)-pattern $\pi = \langle(word(v_1), \ldots, word(v_d)), k\rangle$ with the d-dimensional box $Box(\pi) = I(v_1) \times \cdots \times I(v_d)$ over *Rank*$_{d,k}(S)$. For any set $Q \subseteq [1,n]^d \times [1,m]$, we define the measure of box $Box(\pi)$ by

$$\Delta_{Q,\xi}(Box(\pi)) = count'_1(Q,\pi) - count'_0(Q,\pi),$$

where $count'_\alpha(Q,\pi)$ for $\alpha \in \{0,1\}$ is the number of points $(r_1, \ldots, r_d; i) \in Q$ such that $(r_1, \ldots, r_d; i)$ is in $Box(\pi) \times [1,m]$ and $\xi(s_i) = \alpha$ for $s_i \in S$.

Lemma 3.1
Let S be a sample and $Q = Rank_{d,k}(S)$. Then, for any canonical (d,k)-pattern π, $\Delta_{S,\xi}(\pi) = \Delta_{Q,\xi}(Box(\pi))$.

Proof
This is essentially due to Manber and Baeza-Yates.[10] ∎

Now let v be a node of the suffix tree *Tree*(S). The *level*(v) *of* v *in Tree*(S) is the distance from the root to v. Fig. 2 shows the subprocedure *Discover*. A sequence of points is used as a representation of a set attached to a node of the suffix tree. Also we assume that each node of the suffix tree provides d Boolean flags. By checking the flags and the levels of nodes, we visit all the nodes corresponding to the subwords which possibly appear in patterns.

Lemma 3.2
The subprocedure *Discover*$_{d,k}$ invoked with a sequence Q of points and $c = 1$ generates and measures every canonical (d,k)-pattern for *Tree*(S) that matches at least one document in Q exactly once.

Proof
Assume that *Discover*$_{d,k}$ is invoked with a sequence Q and a parameter $c \leq d$ at the $(c-1)$th level of its recursion. Then clearly the words from the first to the $(c-1)$th of π have already been fixed. We visit and try a node v of *Tree*(S) if and only if the pattern π with its additional cth word *word*(v) can match some

Procedure $Discover_{d,k}(Q, c, \pi)$:

1. If $c \leq d$ then:
   a) Make the new empty queue H.
   b) For each value r of the cth elements of points in Q do:
      i) Make a subsequence Q_r of points whose cth element equals r.
      ii) Attach Q_r to the rth leaf l_r and add l_r to the last of H.
   c) For each node v at the first of the queue H do:
      i) If v is not a leaf, then:
         Let $u_1, \ldots, u_p$ be the children of v.
         Merge all the points in $Q_{u_1}, \ldots, Q_{u_p}$ to Q_w lexicographically with respect to the elements from the $(c+1)$th to the last.
         Discard the sequences $Q_{u_1}, \ldots, Q_{u_p}$.
      ii) Call $Discover(c+1, Q_v, \pi \cdot \langle (word(v)), k \rangle)$, by placing $word(v)$ to the cth word of π, and then clear the flag of v.
      iii) If v is not the root, then:
         If the parent w of v is still not checked and $level(w) + 1 = level(v)$, then check and add w to the last of H.
      iv) Dequeue the first node v from H.
2. If $c = d + 1$ then compute $count_\alpha = |\{\delta \in Q \mid \xi(\delta) = \alpha\}|$ for $\alpha \in \{0, 1\}$.
   If $\Delta := count_1 - count_0$ equals the largest value, then record π in the list of patterns.
   If it is strictly larger than the values seen so far, discard patterns in the list, and add π.

Fig. 2 The Procedure *Discover*

documents in Q. All those documents are prepared in the sequence Q_v of points when v is visited.

Firstly, the leaf l_r of rank r has to be tried as the cth word of a pattern only if Q contains points whose cth elements equal r. Every such leaf is added to H and gets the sequence Q_{l_r} of the points in Step 1-b. Next, any internal node v should be visited if any leaf descended from v is enqueued in H. We will find out all such nodes by passing sequences of points from leaves to the root in Step 1-c-iii. The cth flag of v is set when its child in the 'next level' is visited, and is cleared after v is visited. By induction on the levels, it is clear that if any child in the level $level(v) + 1$ is encountered, then all the children of v in the deeper levels were visited, and all other children in $level(v) + 1$ have already been enqueued in H. So the sequence Q_v supplied to *Discover* in Step 1-c-ii contains only and all the points whose elements from the first to the cth match with $\pi \cdot \langle (word(v)), k \rangle)$.

If $c = d + 1$, the subprocedure $Discover_{d,k}$ computes the measure of the (d, k)-pattern by simply counting labels of points. Therefore, all the canonical patterns are generated and tested by the procedure. ■

Lemma 3.3

Let h be the height of the suffix tree. Then the procedure $Discover_{d,k}$ with an input Q of $\tilde{n}$ points and $c = 1$ runs in the worst case $O(h^d \tilde{n})$ time.

Proof

Let $T_j(\tilde{n})$ for $0 \leq j \leq d$ be the worst-case running time of $Discover_{d,k}$ called with $c = d - j + 1$ (to determine the last j words in the pattern).

Clearly $T_0(\tilde{n})$ runs in $O(\tilde{n})$ time. For $j \geq 1$, Step 1-a takes $O(\tilde{n})$ time, and Step 1-b-i can be done in $O(\log |\Sigma| \cdot |Q_{c,v_i}|)$ time because at most $|\Sigma|$ sorted

sequences have to be merged. Therefore, we have the recurrence

$$T_j(\tilde{n}) = O(\tilde{n}) + \sum_{1 \le i \le \mathit{tree}(S)} (\log|\Sigma| \cdot |Q_{c,v_i}| + T_{j-1}(|Q_{c,v_i}|)),$$

where $\mathit{tree}(S) \le 2n-1$ is the number of nodes of $\mathit{Tree}(S)$. Now we introduce the sets H_l with $0 \le l \le h$ partitioning the nodes of the suffix tree $\mathit{Tree}(S)$ by their levels l. Although levels of leaves can vary, both the sequence of points that will be attached to the root and the union of the sequences attached to the leaves must include all $\tilde{n}$ points supplied in Q. So the above recurrence can be rewritten by the inequality $\sum_{v \in H_l} |Q_{c,v}| \le \tilde{n}$ for all $1 \le l \le h$ to

$$\begin{aligned} T_j(\tilde{n}) &= O(\tilde{n}) + \log|\Sigma| \sum_{1 \le l \le h} \sum_{v \in H_l} |Q_{c,v}| + \sum_{1 \le l \le h} \sum_{v \in H_l} T_{j-1}(|Q_{c,v}|) \\ &\le O(\tilde{n}) + h\tilde{n}\log|\Sigma| + \sum_{1 \le l \le h} \sum_{v \in H_l} T_{j-1}(|Q_{c,v}|). \end{aligned}$$

Solving this recurrence, we have $T_j(\tilde{n}) = O(h^j\tilde{n})$. This completes the proof. ■

Since we have at most $k^{d-1}n$ points in $\mathit{Rank}_{d,k}(S)$, the following holds.

Theorem 3.1
Let (S,ξ) be a pair of sample over Σ and an objective function for S. Then, for fixed integer d and k, our algorithm finds all the (d,k)-patterns in canonical form that maximizes the measure in worst-case complexity $O(k^{d-1}h^dn)$ time and $O(k^{d-1}n)$ space, where n is the total length of strings in S and $h \le n$ is the height of the suffix tree $\mathit{Tree}(S)$.

Corollary 3.1
Let $S \subseteq \Sigma^*$ be a set of uniformly random strings with total length n. Then the problem MAXIMUM AGREEMENT BY d-WORDS k-PROXIMITY ASSOCIATION is solvable in expected time $O(dk^{d-1}n\log^d n)$ and space $O(k^{d-1}n)$, where (S,ξ) is an instance of the problem, and n is the total length of strings in S.

Proof
Applying the observation by Devroye *et al.*[5] on the height of suffix trees for uniformly random strings, we obtain the result. ■

§4 Hardness of Approximating Maximum Agreement by Association of Arbitrary Many Words

As we have seen, MAX AGREEMENT BY d-WORDS k-PROXIMITY ASSOCIATION for fixed d and k is solvable in polynomial time. However, the problem has seemed to be computationally intractable if d and k are not limited. This observation comes from the NP-completeness of a consistency problem of a class of regular patterns[12] in computational learning theory. Although the proof of this result does not capture the hardness of the optimization, it tells us that the maximum agreement problem of word-association patterns is intractable if the optimum is very close to the total number of texts.

Consider the following trivial algorithm: Given an instance, simply count the numbers of positive-labeled strings and negative-labeled strings and choose the better one from either an empty pattern accepting all labeled strings or a pattern rejecting all strings (for example a string longer than all given strings). This algorithm clearly approximates the problem with a guaranteed factor 1/2, and thus MAX AGREEMENT BY WORD ASSOCIATION is in class APX.[2)]

On the other hand, it is hard to approximate the problem within an arbitrarily small error in polynomial time. The main result in this section is the following theorem.

Theorem 4.1
MAX AGREEMENT BY WORD ASSOCIATION is MAX SNP-hard.

Proof
In the following we give a PTAS-reduction from MAX 2-SAT that proves this theorem.

We build an instance $\pi = (S_\phi, \xi_\phi)$ of MAX AGREEMENT BY WORD ASSOCIATION over a finite alphabet A_ϕ for any MAX 2-SAT instance $\phi = (X, C)$, a pair of a set $X = \{x_1, \ldots, x_n\}$ of variables and a set $C = \{c_1, \ldots, c_m\}$ of 2-literal clauses. Firstly, we define an alphabet $A_i = \{a_i, b_i\}$ for every $1 \leq i \leq n+1$ where n is the number of boolean variables $|X|$ in ϕ. Let $bin(i)$ for $1 \leq i \leq n$ be the binary representation of i in $(A_{n+1})^{\lceil \log n \rceil + 1}$ by some natural coding. We denote by symbols $\mathtt{t}_i, \mathtt{f}_i, \mathtt{u}_i, \mathtt{d}_i$ the strings $a_ia_ib_i, b_ia_ia_i, a_ib_ia_i, b_ia_ib_i$ in $(A_i)^3$ for all $1 \leq i \leq n$, respectively. Note that any word-association pattern that accepts both $\mathtt{d}_i$ and $\mathtt{u}_i$ cannot reject either $\mathtt{t}_i$ or $\mathtt{f}_i$, or both. Let S_i for $1 \leq i \leq n$ be a set either $\{\mathtt{t}_i, \mathtt{f}_i, \mathtt{u}_i, \mathtt{d}_i\}$, $\{\mathtt{t}_i, \mathtt{u}_i, \mathtt{d}_i\}$ or $\{\mathtt{f}_i, \mathtt{u}_i, \mathtt{d}_i\}$. Then the notion $[S_i]$ refers a pattern that accepts strings in S_i and rejects $\{\mathtt{t}_i, \mathtt{f}_i, \mathtt{u}_i, \mathtt{d}_i\} - S_i$. For example, $[\{\mathtt{t}_i, \mathtt{d}_i, \mathtt{u}_i\}]$ is a pattern (a_ib_i) or (a_i, b_i), which accepts $\mathtt{t}_i, \mathtt{d}_i, \mathtt{u}$ but rejects $\mathtt{f}_i$.

Next, for each clause $c_i \in C$, we associate the following negative-labeled string: For each j from 1 to n, we concatenate either (i) $\mathtt{d}_j$ if x_j and $\bar{x}_j$ are not in c_i, (ii) $\mathtt{f}_j$ if the literal x_j appears in c_i, or (iii) $\mathtt{t}_j$ if $\bar{x}_j$ is in c_i. Then to its end we append $bin(i)$. For example, we associate with a clause $c_i = (x_j, \bar{x}_k)$ with $1 < j < k < n$ the string $\mathtt{d}_1 \cdots \mathtt{f}_j \cdots \mathtt{t}_k \cdots \mathtt{d}_n \cdot bin(i)$. Note that we are assuming no clause in C has two complementary literals. Such a clause can be handled as the auxiliary additional value to the measure in the reduction.

For each clause c_i, we associate two positive-labeled strings $\mathtt{d}_1 \cdots \mathtt{d}_n \cdot bin(i)$ and $\mathtt{u}_1 \cdots \mathtt{u}_n \cdot bin(i)$. We refer to these by *down-string* and *up-string*, respectively. The labeled sample corresponding to ϕ is defined as the set of the negative-labeled strings and the positive-labeled strings associated with all clause in C.

Now we specify the translation from a solution p for $(A_\phi, T_\phi, \xi_\phi)$ to a Boolean assignment B_p for ϕ. Let S_i for $1 \leq i \leq n$ be either $\{\mathtt{t}_i, \mathtt{u}_i, \mathtt{d}_i\}$ or $\{\mathtt{f}_i, \mathtt{u}_i, \mathtt{d}_i\}$. Then, a *boolean assignment-pattern p for* $(A_\phi, T_\phi, \xi_\phi)$ is a word-association pattern $[S_1] \cdot [S_2] \cdot \cdots \cdot [S_n] \cdot [A_{n+1}^{\lceil \log_2 n \rceil + 1}]$, where $[A_{n+1}^{\lceil \log_2 n \rceil + 1}]$ denotes a pattern that accepts all strings in $A_{n+1}^{\lceil \log_2 n \rceil + 1}$. By regarding $\{\mathtt{t}_i, \mathtt{u}_i, \mathtt{d}_i\}$ and $\{\mathtt{f}_i, \mathtt{u}_i, \mathtt{d}_i\}$ as 'true' and 'false' assignment to x_i respectively, an assignment pattern rejects a negative-labeled string if and only if the corresponding clause is satisfied. This fact is important since the following lemma holds: ■

Lemma 4.1
For any solution p of $\pi = (A_\phi, T_\phi, \xi_\phi)$, we can compute in logspace an assignment-pattern p' for π whose measure is larger than that of p.

We show a brief outline of a proof of this lemma. Without loss of generality, we can suppose that a solution p accepts both some up-strings and down-strings and achieves the measure larger than $2/3 \cdot |T_\phi|$. Otherwise we simply take the empty-pattern ( ) as p. Then, such a pattern p can be decomposed into the (possibly empty) subsequences p_i formed from only letters in A_i for all $1 \leq i \leq n+1$, since p must accept all the positive-labeled strings except those rejected by p_{n+1}.

If we modify the last segment $[p_{n+1}]$ to accept more positive-labeled strings, at most one negative-labeled string will be accepted for each pair of newly accepted up-string and down-string ending with $bin(i)$. Thus we can modify p_{n+1} to pattern that allows to accept all positive-labeled strings. Now let p_i be one of the subsequences of p accepting both $\mathtt{t}_i$ and $\mathtt{f}_i$. We replace every such subsequence p_i with either $[\{\mathtt{d}_i, \mathtt{u}_i, \mathtt{t}_i\}]$ or $[\{\mathtt{d}_i, \mathtt{u}_i, \mathtt{f}_i\}]$. Each choice can be done even arbitrarily, and every such replacement at least retains the number of rejected negative-labeled strings. Thus we obtain an assignment-pattern p' from p.

The Lemma 4.1 tells that the measure of the optimum solution of π is given by some assignment pattern, and a boolean assignment-pattern p corresponds to the boolean assignment, say B_p, to X with respect to the formula ϕ. Then, it is immediate that the following claim hold:

Claim 4.2
For every 2-CNF formula ϕ, there holds $opt(\pi) \leq 5 \cdot opt(\phi)$. For every assignment-pattern p, there holds $opt(\phi) - m(\phi, B_p) = opt(\pi) - m(\pi, p)$.

The first statement is true because, for any 2-CNF formula ϕ, always $opt(\phi) \geq \frac{1}{2}|C|$[8] and $opt(\pi) = 2 \cdot |C| + opt(\phi)$ hold. The second statement is clear since with an assignment-pattern all the positive-labeled strings are accepted and thus only the number of rejected negative-strings corresponding to the satisfied clauses makes a difference between $opt(\pi)$ and $m(\pi, p)$.

Corollary 4.1
There is no PTAS for MAX AGREEMENT FOR WORD ASSOCIATION PATTERNS unless P = NP.

§5 Agnostic PAC-Learning

Agnostic PAC-learning[9] is a generalization of the well known PAC-learning model in computational learning theory, where the learner has no information about the structure behind the target function, or training examples may contain arbitrary large percent of noise. Kearns *et al.*[9] show that for any class of polynomial VC-dimension,[9] the polynomial time solvability of the maximum agreement problem and the efficient agnostic PAC-learnability are equivalent.

Unfortunately, the maximum agreement problem is intractable in most cases.[9] Recently, however, the problem is shown to be efficiently solvable for simple but interesting subclasses of geometric patterns such as axis-parallel rectangles.[6] Since proximity word-association patterns obviously have polynomial VC-dimension, we have the next result from Theorem 3.1 and Theorem 3.1.

Corollary 5.1
For every $d, k \geq 0$, the algorithm *Find_Patterns* in Fig. 1 is an efficient agnostic PAC-learning algorithm for the class of (d, k)-patterns.

§6 Conclusion

In this paper, we presented an efficient algorithm for finding proximity word-association patterns in a large collection of unstructured texts, and we also showed that the problem is hard to approximate when the number of subwords in a pattern is not bounded. An implementation of and a preliminary experiments with of a prototype system are reported in companion papers.[3,4]

References

1) Agrawal, R., Imielinski, T. and Swami, A., "Mining Association Rules between Sets of Items in Large Databases," in *Proc. 1993 SIGMOD*, pp. 207–216, 1993.

2) Ausiello, G., Crescenzi, P. and Protasi, M., "Approximate Solution of NP Optimization Problems," *Theor. Comput. Sc., 150*, pp. 1–55, 1995.

3) Arimura, H., Wataki, A., Fujino, R. and Arikawa, S., "A Fast Algorithm for Discovering Optimal String Patterns in Large Text Databases," in *Proc. the 9th Int. Workshop on Algorithmic Learning Theory, LNAI 1501*, pp. 247–261, 1998.

4) Arimura, H., Kasai, T., Wataki, A., Fujino, R., Shimozono, S. and Arikawa, S., "An Efficient Tool for Discovering Simple Combinatorial Patterns from Large Text Databases," in *Proc. the 1st Discovery Science, LNAI 1532*, pp. 393–394, 1998.

5) Devroye, L., Szpankowski, W. and Rais, B., "A Note on the Height of the Suffix Trees," *SIAM J. Comput., 21*, pp. 48–53, 1992.

6) Dobkin, D. P. Gunopulos, D. and Maass, W., "Computing the Maximum Bichromatic Discrepancy, with Applications to Computer Graphics and Machine Learning," *J. Comut. Sys. Sci., 52*, pp. 453–470, 1996.

7) Fukuda, T. Morimoto, Y., Morishita, S. and Tokuyama, T., "Data Mining Using Two-Dimensional Optimized Association Rules," in *Proc. 1996 SIGMOD*, pp. 13–23, 1996.

8) Johnson, D. S., "Approximation Algorithms for Combinatorial Problems," *J. Comut. Sys. Sci., 9*, pp. 256–278, 1974.

9) Kearns, M. J., Shapire, R. E. and Sellie, L. M., "Toward Efficient Agnostic Learning," *Machine Learning, 17*, pp. 115–141, 1994.

10) Manber, U. and Baeza-Yates, R., "An Algorithm for String Matching with a Sequence of Don't Cares," *Inf. Procc. Lett., 37*, pp. 133–136, 1991.

11) McCreight, E. M., "A Space-Economical Suffix Tree Construction Algorithm," *J. ACM, 23*, pp. 262–272,1976.

12) Miyano, S., Shinohara, A. and Shinohara, T., "Which Classes of Elementary Formal Systems are Polynomial-Time Learnable," in *Proc. 2nd Workshop on Algorithmic Learning Theory*, pp. 139-150, 1991.

13) Papadimitriou, C. H. and Yannakakis, M., "Optimization, Approximation, and Complexity Classes," *J. Comput. Sys. Sci.*, *43*, pp. 425–440, 1991.

14) Wang, J. T.-L., Chirn, G.-W., Marr, T. G. , Shapiro, B., Shasha, D. and Zhang., K., "Combinatorial Pattern Discovery for Scientific Data: Some preliminary results," in *Proc. 1994 SIGMOD*, pp. 115–125, 1994.

Shinichi Shimozono, Ph.D.: He is an Associate Professor of the Department of Artificial Intelligence at Kyushu Institute of Technology Iizuka, Japan. He obtained the B.S. degree in Physics from Kyushu University, awarded M.S. degree from Graduate School of Information Science in Kyushu University, and his Dr.Sci. degree in 1996 from Kyushu University. His research interests are primarily in the design and analysis of algorithms for intractable problems.

Hiroki Arimura, Ph.D.: He is an Associate Professor of the Department of Informatics at Kyushu University, Fukuoka, Japan. He is also a researcher with Precursory Research for Embryonic Science and Technology, Japan Science and Technology Corporation (JST) since 1999. He received the B.S. degree in 1988 in Physics, the M.S. degree in 1979 and the Dr.Sci. degree in 1994 in Information Systems from Kyushu University. His research interests include data mining, computational learning theory, and inductive logic programming.

Setsuo Arikawa, Ph.D.: He is a Professor of the Department of Informatics and the Director of University Library at Kyushu University, Fukuoka, Japan. He received the B.S. degree in 1964, the M.S. degree in 1966 and the Dr.Sci. degree in 1969 all in Mathematics from Kyushu University. His research interests include Discovery Science, Algorithmic Learning Theory, Logic and Inference/Reasoning in AI, Pattern Matching Algorithms and Library Science. He is the principal investigator of the Discovery Science Project sponsored by the Grant-in Aid for Scientific Research on Priority Area from the Ministry of ESSC, Japan.