Pattern Matching with Taxonomic Information

Takuya Kida
Hokkaido University
Kita 14, Nishi 9, Kita-ku,
Sapporo, 060-0814, Japan
kida@ist.hokudai.ac.jp

Hiroki Arimura
Hokkaido University
Kita 14, Nishi 9, Kita-ku,
Sapporo, 060-0814, Japan
arim@ist.hokudai.ac.jp

ABSTRACT

In this paper, we study the pattern matching problem with taxonomic information (PMTX, for short), where taxonomy is a partially ordered set of letters describing an IS-A hierarchy. We present an efficient algorithm for PMTX that runs in $O(\frac{mn}{w})$ time with $O(m+\frac{mh}{w})$ preprocess and $O(\frac{m\sigma}{w})$ extra space, where h, m, t, σ, and w are the size of the taxonomic information $\mathcal{H}$, the length of the pattern $P \in \Sigma^*$, the length of the text $T \in \Sigma^*$, the cardinality of the sorted alphabet Σ, and the computer word length, respectively. If the pattern length m is less than w, it runs in $O(n)$ time, $O(m + h)$ preprocess, and $O(\sigma)$ space, and works very fast in practice. We also discuss various improvements of the algorithms for real world applications.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search process*

General Terms

Algorithms

Keywords

string matching, taxonomy, bit-parallelism

1. INTRODUCTION

Information retrieval is one of the basic means for extracting useful information from massive collections of text data. In particular, *pattern matching* studies the design and the analysis of efficient algorithms for textual problems arising in information retrieval and may provide theoretical basis for implementing fast information retrieval systems with large text data.

Traditionally, the study of pattern matching has concentrated on fast search by ignoring any background information. On the other hand, a significant amount of domain knowledge for textual information is becoming available online in the form of thesaurus or taxonomy [6, 5] recently. Potential demands for more intelligent text search that can incorporate with these knowledge are increased [9].

In this paper, we study the pattern matching problem with taxonomic information (PMTX, for short). A *taxonomy* is a pair $\mathcal{H} = (\Sigma, \succeq)$ of an alphabet Σ of terms, or *concepts*, and a partial order $\succeq$ describing an 'IS-A' hierarchy among concepts. For example, Figure 1 shows an example of biological taxonomy from Gene Ontology [5], which contains 16,675 terms as of June 2004. A pattern and a text over the taxonomy are concept sequences, i.e., string of concepts of Σ. Given a taxonomy $\mathcal{H} = (\Sigma, \succeq)$, a symbol A matches another symbol B if A appears above B in the taxonomy, i.e., $A \succeq B$. Then, pattern P of length m matches text T of length n if there exists a consecutive substring S of T such that $P[i]$ matches $S[i]$ for every $i = 1, \ldots, m$, where $S[i]$ denotes the i-th symbol of a string S. Figure 2 shows an example of patterns and texts under the biological taxonomy of Figure 1, where a general pattern "`(cell) (receptor) (for) (catalytic activity)`" matches a more specific substring "`(cell surface) (receptor) (for) (hyaluronate)`" of the title line.

```
!autogenerated-by:     DAG-Edit version 1.417
!saved-by:             gocvs
!date:                 Fri Jun 11 23:15:37 PDT 2004
!version: $Revision: 3.60 $
!type: % is_a is a; !type: < part_of Part of
$Gene_Ontology; GO:0003673
 %cell; GO:0005623
   %insoluble fraction; GO:0005626
   %membrane fraction; GO:0005624
    %vesicular fraction; GO:0042598
     %microsome; GO:0005792
   %cell surface; GO:0009928
   %cell envelope; GO:0030313
   %cell wall; GO:0005618  ...
 %molecular_function; GO:0003674
   %catalytic activity; GO:0003824
    %lyase activity; GO:0016829
     %hyaluronate; hyaluronate lyase activity;GO:0030340
```

Figure 1: An example of biological taxonomy from Gene Ontology

The first six lines starting with '!' are comments. Each string between '%' and ';' stands for a term or a concept name, and indents represent the nesting level. The numbers prefixed with 'GO:' are id's for concepts and any descriptions related to the 'part-of' relation are not considered here.

PMTX is a generalization of the matching problems for string patterns, string patterns with wildcards, patterns with symbol-classes. Although a straightforward algorithm solves the PMTX problem in $O(mn)$ time and $O(|\Sigma|^2)$ space, real world applications in large text databases requires more efficient algorithms.

Our main contribution is an efficient algorithm that solves PMTX in $O(\frac{mn}{w})$ running time with $O(m + \frac{mh}{w})$ prepro-

(a) Pattern P

```
(cell) (receptor) (for) (catalytic activity)
```

(b) Text T

Publ:1: Cell. 1990 Jun 29;61(7):1303-13.
Title:CD44 is the principal (cell surface) (receptor) (for) (hyaluronate).
Authors:Aruffo A, Stamenkovic I, Melnick M, Underhill CB, Seed B.
Abst:CD44 is a broadly distributed cell surface (protein) thought to mediate (cell) attachment to (extracelular matrix components) or specific (cell surface) (ligands). We have created soluble CD44- (immunoglobulin fusion proteins) and characterized their reactivity with tissue sections and lymph node high endothelial cells in primary culture. ...
PMID:1694723 [PubMed - indexed for MEDLINE]

Figure 2: An example of biological patterns and texts under the taxonomy

The text is an entry from PubMed [8]. A parenthesized string indicates a concept and an underline indicates the match, where concepts '*cell*' matches '*cell surface*' and '*catalytic activity*' matches '*hyaluronate*'.

cessing and $(\frac{m|\Sigma|}{w})$ extra space on a random access machine (RAM) with bit-width $w = \log n$ (say, 32 or 64 in a current system), where h, $|\Sigma|$, m, and n are the size of the taxonomic information $\mathcal{H}$, the cardinality of the taxonomy Σ, the length of the pattern $P \in \Sigma^*$, and the length of the text $T \in \Sigma^*$, respectively.

The algorithm is based on the *bit-parallel technique*, proposed by Abrahamson [1], later extended by [4, 10], which accelerates a pattern matching algorithm by the factor of w by performing bit operations in $O(1)$ time. Although simple application of this technique yields an $O(\frac{mn}{w})$ time algorithm with $O(m \cdot h)$ preprocessing and $O(\frac{m|\Sigma|}{w})$ space, we improve this to an algorithm with $O(m + \frac{mh}{w})$ preprocessing using dynamic programming over $\mathcal{H}$. As a corollary, when the length of a pattern is less than w, we can solve PMTX in $O(n)$ running time with $O(m+h)$ preprocessing and $O(|\Sigma|)$ space, and thus this is optimal in time complexity.

Finally, we discuss the case where concepts are encoded by strings on a fixed alphabet, as in Figure 1, e.g., on ASCII alphabet. We also discuss the practical improvements on PMTX with many terminal symbols. Possible applications of our algorithm for PMTX are genome sequence search with symbol taxonomy, and biological text search with term taxonomy. Furthermore, foreign language search with translation dictionary can be an application of this framework. The taxonomies considered in this paper can be viewed as a primitive form of ontology [9] without structural information. Thus, the results presented in this paper will be the basis of the further studies on efficient pattern matching algorithms with more complicated ontology structures.

This paper is a short version of [7].

2. PRELIMINARIES

2.1 Sorted Alphabet

We introduce a sorted alphabet as a model of taxonomic information. Let Σ be a finite *alphabet*. An element of Σ is called a *symbol* or a *concept*. Let binary relation $\succeq$ be a partial order on Σ. A sorted alphabet is a partially ordered set $(\Sigma, \succeq)$. Given a sorted alphabet $(\Sigma, \succeq)$, for any $c, d \in \Sigma$, we say that c *matches* d if $c \succeq d$. In the following, we say Σ for a sorted alphabet $(\Sigma, \succeq)$ if no confusion occurs. For $a, b \in \Sigma$, we define $a \succ b$ if $a \succeq b$ and $a \neq b$, and $a \gg b$ if there is no $x \in \Sigma$ such that $a \succ x \succ b$. We also denote by $Par(a)$ the set $\{b \in \Sigma \mid b \gg a\}$ for $a \in \Sigma$.

A sorted alphabet is represented as a *Hasse diagram*. A Hasse diagram for $(\Sigma, \succeq)$ is the direct acyclic graph (DAG) $\mathcal{H} = (V_{\mathcal{H}}, E_{\mathcal{H}})$ defined as follows: $V_{\mathcal{H}} = \Sigma$ is the set of vertices. For each $a, b \in \Sigma$, $\mathcal{H}$ has the edge (a, b) iff $a \gg b$. For a set A, $|A|$ denotes the cardinality of A. We define the size of $\mathcal{H}$ as $|\mathcal{H}| = |V_{\mathcal{H}}| + |E_{\mathcal{H}}|$. We define for $a \in \Sigma$ the sets $Upb(a) = \{b \in \Sigma \mid b \succeq a\}$ and $Lob(a) = \{b \in \Sigma \mid a \succeq b\}$.

An element $T = a_1 \cdots a_n (n \geq 0)$ of Σ^* is called a *string* or a *concept sequence*. Then, the length of the string T is denoted by $|T| = n$. For $1 \leq i \leq |T|$, the i-th symbol of a string T is denoted by $T[i] = a_i$, and for $1 \leq i \leq j \leq |T|$, the substring of T that begins at position i and ends at position j is denoted by $T[i : j] = a_i \cdots a_j$.

2.2 Pattern Matching on Sorted Alphabet

Let $P \in \Sigma^*$ and $T \in \Sigma^*$ be a *pattern* and a *text*, respectively. Then, we say that P matches T at position $\ell\,(1 \leq \ell \leq |T| - |P| + 1)$ iff $P[i] \succeq T[i + \ell - 1]$ for all $i = 1, \ldots, |P|$. If P matches T at position ℓ, such ℓ is called an *occurrence* of P in T. In this paper, we assume that taxonomic information is given as a sorted alphabet. The pattern matching problem with taxonomic information can be stated as follows.

Pattern Matching Problem
with Taxonomic Information (PMTX)
Input: A sorted alphabet $(\Sigma, \succeq)$ represented by the Hasse diagram $\mathcal{H}$, a string $P \in \Sigma^*$ as a pattern, and a string $T \in \Sigma^*$ as a text.
Output: All occurrences of P in T.

2.3 Pattern Matching with Bit-parallelism

A pattern matching algorithm that allows each pattern position to match a *set* of symbols rather than a single symbol, was first proposed by Abrahamson [1]. This type of algorithm can deal with range of symbols (e.g., "a-z"), complements (e.g. any symbol except a number), and wild cards. Baeza-Yates & Gonnet [4] extended the algorithm so that it can handle k-mismatches. Wu & Manber [10] also extended it later into the approximate string matching prob-

lem, which can handle insertions or deletions in addition to mismatches. These algorithms are based on the bit-parallel technique, and work fast in practice. We present below the Shift-And algorithm according to the notation in [1].

Let Δ be an unsorted alphabet. Let $P \in \Delta^*$ be a pattern of length m, and $T \in \Delta^*$ be a text of length n. Then, for $k = 0, 1, \ldots, n$, we want to compute the set

$$R_k = \{1 \le i \le m \mid i \le k \text{ and } P[1:i] = T[k-i+1:k]\}.$$

If $m \in R_k$, then $T[k-m+1:k] = P$, namely, there is an occurrence $\ell = k - m + 1$. To compute R_k, we define for $a \in \Delta$ the set

$$M(a) = \{1 \le i \le m \mid P[i] = a\}.$$

For a set A of integers and an integer k, let $A \oplus k = \{i + k \mid i \in A\}$, and define the function f as follows.

DEFINITION 1. *Define the function* $f : 2^{\{1,2,\cdots,m\}} \times \Delta \to 2^{\{1,2,\cdots,m\}}$ *by*

$$f(S, a) = \big((S \oplus 1) \cup \{1\}\big) \cap M(a),$$

where $S \subseteq \{1, \cdots, m\}$ *and* $a \in \Delta$.

By using this function, we can compute R_k recursively for $k = 1, 2, \ldots, n$ by

1. $R_0 = \emptyset$,
2. $R_{k+1} = f(R_k, T[k+1]) \qquad (k \ge 0)$.

The pattern matching process is as follows: For $k = 1, 2, \ldots, n$, the algorithm reads $T[k]$ and computes R_k recursively by using f, and then examines whether m is in R_k.
By representing the sets R_k and $M(a)$ as m-bit integers, the above recursive formulas can be transformed as follows.

1. $R_0 = 0$,
2. $R_{k+1} = ((R_k \ll 1) + 1)\ \&\ M(T[k+1]) \qquad (k \ge 0)$,

where '$\ll$' and '&' denote the bit-shift operation and the bit-wise logical product operation, respectively. Then, pattern occurrences can be detected if $R_k \& 2^{m-1} \neq 0$.
The algorithm scans the text T in $O(\frac{mn}{w})$ time if the sets M are computed. If $m \le w$ and bit-wise logical operations can be done in $O(1)$ time, the scanning time reduces into $O(n)$.

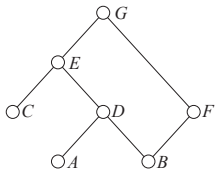
3. PROPOSED ALGORITHM

3.1 Overview of Algorithm

In order to apply the Shift-And algorithm stated in Section 2.3 to our problem, we extend the definition of M. Let Σ be a sorted alphabet. Define the mask table $M'(a)$ for any $a \in \Sigma$ as:

$$M'(a) = \{1 \le i \le m \mid P[i] \succeq a\}.$$

The text scanning process is the same as the ordinary case of the Shift-And algorithm. Thus it can be done in $O(\frac{mn}{w})$ time after M' is computed in preprocessing. The remained problem is how to compute the table M'. From here we concentrate the computation of it. The following lemma characterizes that the above extension leads the pattern matching problem allowing a set of symbols.



$P =$	A	B	C	D	E	F
$M(a)$	1	2	3	4	5	6
A	√			√	√	
B		√		√	√	√
C			√		√	
D				√	√	
E					√	
F						√
G						

Figure 3: $M'(a)$ for pattern $P =$ ABCDEF.
The symbol √ indicates that $i \in M'(a)$ for $a \in \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{G}\}$.

```
1. Initialization M(a):
2.     M(a) = {1 ≤ i ≤ m | P[i] = a};
3. For each a ∈ Σ do begin
4.     Call CalculateM'(a);
5. end

6. Function CalculateM'(a)
7. begin
8.     If M'(a) has already calculated then return M'(a);
9.     else do begin
10.        M'(a) = M(a);
11.        for each x ∈ Par(a) do begin
12.            M'(a) = M'(a)∪ (CalculateM'(x));
13.        end;
14.    end;
15.end
```

Figure 4: Procedure to calculate $M'(a)$.

LEMMA 1. *Let* $(\Sigma, \succeq)$ *be a given sorted alphabet, and* $P \in \Sigma^*$ *be a given pattern. For any* $a \in \Sigma$*, we obtain the formula:*

$$M'(a) = \bigcup_{x \in Upb(a)} M(x).$$

For example, for $P =$ ABCDE and Σ displayed at the left of Figure3, the table at the right of Figure3 indicates the above M'. Although we can fill the table in $O(m|\mathcal{H}|)$ time from Lemma 1, by traversing $\mathcal{H}$ and checking whether $P[i] \succeq a$ for each $P[i]$, the time complexity can be reduced further.

Next, we will show how to compute M' in $O(m + \frac{m|\mathcal{H}|}{w})$ time below. The following lemma is a key for reducing the time complexity.

LEMMA 2. *Let* Σ *be a given sorted alphabet and* $P \in \Sigma^*$ *be a pattern. For any* $a \in \Sigma$*, we have the following formula:*

$$M'(a) = M(a) \cup \bigcup_{x \in Par(a)} M'(x).$$

From Lemma 2, the mask table M' can be computed recursively by traversing the DAG $\mathcal{H}$. The procedure to compute M' is illustrated in Figure4. The line 2 in Figure4 takes $O(m)$ time. The procedure is called $O(|\mathcal{H}|)$ times rather than $O(|\Sigma|)$ times. Note that, however, it becomes $O(|\Sigma|)$ if $\mathcal{H}$ is a tree. Since we can compute the union of sets in $O(\frac{m}{w})$ time by representing them as integers, the time complexity of the calculation of M' is totally $O(\frac{m|\mathcal{H}|}{w})$. Therefore, we obtain the following lemma.

LEMMA 3. *Let $\mathcal{H}$ be the DAG for $(\Sigma, \succeq)$ and $P \in \Sigma^*$ be a pattern. Then, the mask table M' can be computed in $O(m + \frac{m|\mathcal{H}|}{w})$ time.*

Since $M'(a)$ for each $a \in \Sigma$ is stored as an integer, the mask table M' can be stored totally in $O(\frac{m|\Sigma|}{w})$ space. From the above discussion, we obtain the followings.

THEOREM 1. *The pattern matching problem with taxonomy information can be solved in $O(\frac{mn}{w})$ time with $O(m + \frac{m|\mathcal{H}|}{w})$ preprocessing and $O(\frac{m|\Sigma|}{w})$ extra space on an arithmetic RAM with word length $w = O(\log n)$.*

COROLLARY 1. *The pattern matching problem with taxonomic information can be solved in $O(n)$ time with $O(m+|\mathcal{H}|)$ preprocessing and $O(|\Sigma|)$ extra space, assuming that the length of pattern is less than w, i.e., $m \leq w$.*

3.2 Improvements for Large Alphabets

Let $L(\Sigma)$ be the set of minimal elements of a taxonomy Σ, i.e., there is no $x \in \Sigma$ for any $a \in L(\Sigma)$ such that $a \succ x$. Define $C(\Sigma) = \Sigma \setminus L(\Sigma)$. In practice, we observe that $|C(\Sigma)|$ is relatively small, while $|L(\Sigma)|$ is huge. For example, a Japanese lexicon, 'Goi-Taikei'[1], $|C(\Sigma)|$ and $|L(\Sigma)|$ are about $3,000$ and $400,000$, respectively. For EDR concept dictionary [6], those are about $6,000$ and $410,000$, respectively.

From this point of view, we can reduce time and space to compute M' in practice, by calculating $M'(a)$ only for $a \in C(\Sigma)$ in the preprocessing stage and then for $a \in L(\Sigma)$ on demand when it happens in the scanning stage.

3.3 Application to Ordinal Texts

Although we regard so far a concept in Σ as a primitive element, it is often the case that it is encoded as a sequence of readable characters on a fixed alphabet. For example, in the example of Gene Ontology of Figure 1, a concept `(cell surface)` is encoded in ASCII code. Below, we discuss how to handle this case efficiently.

Let Δ be a fixed alphabet and $\Sigma_s \subseteq \Delta^*$ be a concept alphabet encoded appropriately in Δ. Now we have a text $T \in \Delta^*$ and a pattern $P \in (\Delta \cup \Sigma_s)^*$. For $a \in \Sigma_s$, we denote by $Str(a)$ the string which a represents, and extend it for the set $A \in \Sigma_s$ in a natural way, that is, $Str(A) = \{Str(x) \mid x \in A\}$. Let $D(P) \subseteq \Sigma_s$ be a set of concepts included in the pattern. Then, what we should do is to find out the occurrences of $Str(\{Lob(x) \mid x \in D(P)\})$, and to replace them with the corresponding concepts. The replace automaton [3], which is the finite state transducer corresponding to AC machine [2], can accomplish this task in $O(h + |T|)$ time with $O(h)$ extra space, where h is the sum of the length of $Str(\{Lob(x) \mid x \in D(P)\})$. Note that $h = O(|\mathcal{H}|)$.

4. CONCLUSIONS

In this paper, we addressed the pattern matching problem with taxonomic information, and presented an efficient algorithm that solves the problem. The algorithm runs in $O(\frac{mn}{w})$ time with $O(m+\frac{mh}{w})$ preprocessing and $O(\frac{m\sigma}{w})$ extra space, using the bit-parallel technique to achieve the speed-up with factor w, where h, m, n, w, and σ are the size of the taxonomic information $\mathcal{H}$, the length of the pattern $P \in \Sigma^*$, the length of the text $T \in \Sigma^*$, the computer word length (say 32 or 64), and the cardinality of the sorted alphabet Σ, respectively. If the pattern length m is less than w, it runs in $O(n)$ time, $O(m + h)$ preprocess, and $O(\sigma)$ space, and works very fast in practice.

In future, we will implement our algorithm and carry out experiments on real world datasets. Although we did not mention here, extensions to various pattern matching problems are possible. Extension to approximate matching or VLDC matching would be an easy exercise by using the method of [10].

An interesting extension will be the use of a thesaurus as well as a taxonomy in pattern matching. However, since thesaurus contains strings of various length allowing several overlapping, such pattern matching is not an easy task. Moreover, it becomes important to combine thesaurus and taxonomy for more intelligent text search systems.

[1] Please refer `http://www.kecl.ntt.co.jp/icl/mtg/topics/1997/shirai-e.html`

5. REFERENCES

[1] Abrahamson, K. Generalized string matching. *SIAM J. Comput.*, 16(6):1039–1051, December 1987.

[2] Aho, A. V. and Corasick, M. J. Efficient string matching: An aid to bibliographic search. *Comm. ACM*, 18(6):333–340, 1975.

[3] Arikawa, S. and Shiraishi, S. Pattern matching machines for replacing several character strings. *Bulletin of Informatics and Cybernetics*, 21(1–2):101–111, 1984.

[4] Baeza-Yaltes, R. and Gonnet, G. H. A new approach to text searching. *Comm. ACM*, 35(10):74–82, October 1992.

[5] Gene Ontology Consortium. Gene ontology: Tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000. `http://www.geneontology.org/`.

[6] Japan Electronic Dictionary Research Institute, Ltd. EDR Electronic Dictionary Technical Guide (2nd edition). Technical Report TR-045, 1995. `http://www.iijnet.or.jp/edr`.

[7] Kida, T. and Arimura, H. Pattern matching with taxonomic information. Computer Science Division, Knowledge Software Science Group, Gradutate School of Information Science and Technology, Hokkaido University, manuscript, 2004.

[8] NCBI. PubMed 2004. `http://www.ncbi.nlm.nih.gov/PubMed/`.

[9] Stevens, R., Horrocks, I., Goble, C., and Bechhofer, S. Building a reson-able bioinformatics ontology. *IEEE Transactions on Information Technology and Biomedicine*, 6(2):136–41, 2002.

[10] Wu, S. and Manber, U. Fast text searching allowing errors. *Comm. ACM*, 35(10):83–91, October 1992.