HAKKE: A Multi-Strategy Prediction System for Sequences

Naohiro Furukawa [1] Satoshi Matsumoto [1] Ayumi Shinohara [1]
Takayoshi Shoudai [1] Satoru Miyano [2]
{furukawa, matumoto, ayumi, shoudai}@i.kyushu-u.ac.jp
miyano@ims.u-tokyo.ac.jp

[1] Department of Informatics, Graduate School of Information Science and Electrical Engineering, Kyushu University
6-10-1 Hakozaki, Higashi-ku, Fukuoka 812-81 Japan

[2] Human Genome Center, Institute of Medical Science, University of Tokyo
4-6-1 Shirokanedai, Minato-ku, Tokyo 108 Japan

Abstract

We developed a machine learning system HAKKE which is suitable for predicting functional regions from sequences, such as protein-coding region prediction, and transmembrane domain prediction. HAKKE is a hybrid system cooperated by a number of algorithms of a pool to make an accurate prediction. The system uses an extension of the weighted majority algorithm in order to fit the strength of each algorithm into given training examples. In this paper, we describe the core of the system and show some experimental results on transmembrane domain and α-helix predictions.

1 Introduction

Databases of DNA and amino acid sequences compile many sequences where some regions of specific functions or structures are indicated as segments. For example, α-helices, β-sheets, transmembrane domains, exons and signal peptides are specified by positions on the sequences. In this paper we call such specified regions as *marked regions*.

For each family of sequences with marked regions of a specific type, some programs have been developed for predicting such marked regions on unknown sequences. For example, Chou-Fasman's method [1] is well known for predicting the α-helices and β-sheets. The hydropathy

plot of Kyte and Doolittle [2] predicts transmembrane domains on amino acid sequences of membrane proteins. For coding region prediction in DNA sequences, GRAIL [5] and GRAIL II [6] are known as successful systems.

Instead of designing a prediction program individually for a specific family of sequences, we are developing a universal prediction system HAKKE for general use. HAKKE is a hybrid system which cooperates with algorithms of a pool and produces a prediction algorithm by employing an extension of the weighted majority (WM) algorithm in [3] for adjusting the weights of algorithms. HAKKE assumes a pool of algorithms for prediction. The algorithms in the pool are provided in two ways. First, given a sample collection of sequences with marked regions, HAKKE has an ability to automatically generate candidate algorithms which may predict the marked regions of the sequences. In addition to these candidate algorithms generated by HAKKE, users can add into the pool some *ad hoc* algorithms based on their domain knowledge on the sequences.

The advantage of HAKKE is three folds. The first is the simple use of the system. For an unknown sequence which may belong to some family of sequences with marked regions, a user simply needs to supply a sample collection of sequences with marked regions in the family together with the unknown sequence. Then HAKKE makes a prediction of marked regions on the unknown sequence and also produces a program for this prediction that may be used for further prediction.

The second is the possibility of finding a better predictor. The WM algorithm has a possibility to produce a correct prediction algorithm even if none of the prediction algorithms in the pool is exactly correct. Our WM algorithm allows "I don't know" answer in addition to "yes" and "no" if a prediction algorithm does not have enough confidence. Some experiments show that the use of abstention in the voting system is helpful to make good prediction algorithms. The current version of HAKKE employs BONSAI [4] for generating candidate prediction algorithms from sample sequences. Although BONSAI is not a universally powerful system, it succeeded to find good knowledge in its experiments [4] and therefore the prediction algorithms based on this acquired knowledge have a chance to be good ones. Furthermore, *ad hoc* prediction algorithms, which are recognized as good predictors, can be enrolled in the pool of the prediction algorithms. Since HAKKE hybridizes these *ad hoc* prediction algorithms with a large number of prediction algorithms generated by BONSAI by the weighted majority voting with abstention, the prediction algorithm produced by HAKKE would be at least as good as each prediction algorithm in the pool and is likely to achieve more accurate prediction power.

The third is that HAKKE works as a knowledge discovery system. Prediction algorithms generated by BONSAI are constructed from decision trees over regular patterns and alphabet indexings [4]. Therefore we can regard each prediction algorithm as a knowledge representation of the data. The program produced by HAKKE shows which prediction algorithms are important by their weights. Thus HAKKE involves a process of knowledge discovery.

This paper describes an overview of HAKKE system and show some experimental results performed by a prototype of the system. The current version of HAKKE is not equipped with *ad hoc* algorithms. The preliminary experiments on transmembrane domain prediction and α-helix prediction convinced us that the system is enough practical and useful.

2 Preliminaries

In the literature, many efforts have been made to develop a prediction system for a specific family of proteins. For example, the sequence in Figure 1 is an amino acid sequence of a protein where the underlined segments are α-helices. The α-helix prediction problem is to determine the segments for α-helices on unknown sequences.

MNIFEMLRIDEGLRLKIYKDTEGYYTIGIGHLLTKSPSLNAAKGELDKAI
GRNTNGVITKDEAEKLFNQDVDAAVRGILRNAKLKPVYDSLDAVRRAALI
NMVFQMGETGVAGFTNSLRMLQQKRWDEAAVNLAKSRWYNQTPNRAKRVI
TTFRTGTWDAYK

Figure 1: The underlined segments are α-helices.

The main concern of this paper is to cope with such problems in a general framework. This section provides a terminology for this general framework.

Let Σ be a finite alphabet. We call elements in Σ^* *strings* or *sequences*. We use both names for calling the elements in Σ^*. For a string s in Σ^*, we denote by $s[i]$ the i-th character of s, and by $s[i..j]$ the substring $s[i]s[i+1]\cdots s[j]$.

Definition 1. Let $\mathbf{B} = \{0, 1\}$. A pair $e = (s, m)$ of strings s in Σ^+ and m in $\mathbf{B}^+$ with $|s| = |m|$ is called a *marked example*. The string m is called a *marking* of e. A maximal nonempty substring $s[i..j]$ of s marked with 1 is called a *marked region* of s, i.e., $m[i..j]$ is in $\{1\}^+$ and no $m[i-k..j+l]$ with $k+l \geq 1$ is in $\{1\}^+$. In a similar way, an *unmarked region* of s is defined as a maximal nonempty substring $s[i..j]$ of s marked with 0.

Example 1. Let (s, m) be a marked example, where $s =$ `aabbbabaab` and $m = 0111001100$. Then substrings $s[2..4] =$ `abb` and $s[7..8] =$ `ba` are marked regions and $s[1] =$ `a`, $s[5..6] =$ `ba` and $s[9..10] =$ `ab` are unmarked regions. On the other hand, the substring $s[3..6]$ is neither a marked region nor an unmarked region.

A mapping $\varphi : \Sigma^+ \to \mathbf{B}^+$ is called a *marking* on Σ^+ if $|\varphi(s)| = |s|$ for all s in Σ^+. A marking defines the set $E(\varphi) = \{(s, \varphi(s)) \mid s \in \Sigma^+\}$ of marked examples.

Informally, a predictor generator is a program which produces from marked examples of an unknown marking φ a program realizing the marking φ.

3 Weighted Majority Algorithm

This section reviews the idea of the *weighted majority algorithm* (WM, for short) by Littlestone and Warmuth [3] and presents an idea of extending WM for our HAKKE system.

WM assumes a pool of prediction algorithms, each of which answers 0 or 1 for any question. WM is a kind of *master algorithm* [3] that uses the prediction algorithms of the pool to learn a prediction algorithm. Initially, WM assigns a positive weight to each algorithm of the pool.

WM makes its prediction by weighted majority voting of the algorithms. When the prediction of WM was incorrect, WM gives a penalty to each algorithm which voted incorrectly: the penalty is done by decreasing the weight by multiplying a fixed real number β ($0 \leq \beta < 1$) called the *penalty parameter*. If $\beta > 0$, WM gradually decreases the influence of algorithms which made more mistakes and gives relatively high weights to the algorithms which made less mistakes. In case that $\beta = 0$, WM ignores the algorithms of the pool that are inconsistent with given examples, since the all weights of the algorithms which have ever incorrectly voted are set to be zero. In this way, WM constructs a prediction algorithm by determining the weights of the algorithms of the pool.

For the design of HAKKE, we extend WM in two ways. The first extension is due to a practical demand. We separate the penalty parameter β into β_0 and β_1: We use β_0 for the case that the prediction of WM is 0 but the correct answer is 1. On the other hand, β_1 is used for the case that the prediction of WM is 1 but the correct answer is 0. The aim of these parameters is to distinguish the accuracy for marked regions from that for unmarked regions, since the ratio of marked regions to unmarked regions varies widely according to the data. By selecting β_0 and β_1 appropriately, we can control the behavior of the predictor produced by the system.

The second extension is an introduction of "abstention" in the voting. Namely, each algorithm in the pool is permitted to answer "I do not know" to the questions for which it has little confidence. Thus the answers of the algorithms are chosen from $\{0, 1, *\}$, where the symbol $*$ means "abstention". If the prediction of WM was incorrect, the weights of algorithms which made abstention are each multiplied by a fixed real number γ. A weaker penalty is given to abstention from voting, i.e., $0 \leq \beta_0, \beta_1 < \gamma \leq 1$. If all algorithms of the pool answers $*$, then our extended weighted majority algorithm makes a default prediction, 0 or 1. We denote by WM* the weighted majority algorithm extended in this way.

Remark 1. The majority voting strategy with abstention has the following advantages. First, there are some cases that the majority voting by WM increases its prediction ability. For example, let $\Sigma = \{a, b, c\}$ and $\varphi : \Sigma \to \{0, 1\}$ be a function defined by $\varphi(a) = \varphi(b) = \varphi(c) = 1$. The pool of prediction algorithms consists of functions f_1, f_2 and f_3 such that $f_1(a) = f_1(b) = 1$, $f_1(c) = 0$, $f_2(b) = f_2(c) = 1$, $f_2(a) = 0$, and $f_3(a) = f_3(c) = 1$, $f_3(b) = 0$. Then WM predicts exactly the same function as φ when weights $w_1 = w_2 = w_3 = 1$ are given to the prediction algorithms f_1, f_2 and f_3, respectively. On the other hand, $\varphi \neq f_i$ for $1 \leq i \leq 3$.

Remark 2. The use of abstention also increases the prediction ability. Let $\Gamma = \{a, b\}$ and let $\psi : \Gamma \to \{0, 1\}$ be a function defined by $\psi(a) = \psi(b) = 1$. The pool consists of the functions f and g be such that $f(a) = g(b) = 1$ and $f(b) = g(a) = 0$. Note that WM cannot produce the same function as ψ for any weights on f and g. By introducing "abstention", we divide f into two functions f_0 and f_1 such that $f_0(a) = 1$, $f_0(b) = *$, $f_1(a) = *$ and $f_1(b) = 0$. In the same way, g is divided into g_0 and g_1 such that $g_0(a) = 0$, $g_0(b) = *$, $g_1(a) = *$ and $g_1(b) = 1$. Then we can verify that the majority voting with abstention correctly predicts ψ for arbitrary initial weights on the functions in the pool.

4 HAKKE System

In this section, we give an overview of the system HAKKE and describe how HAKKE determines marked regions of unknown sequences (Figure 2).

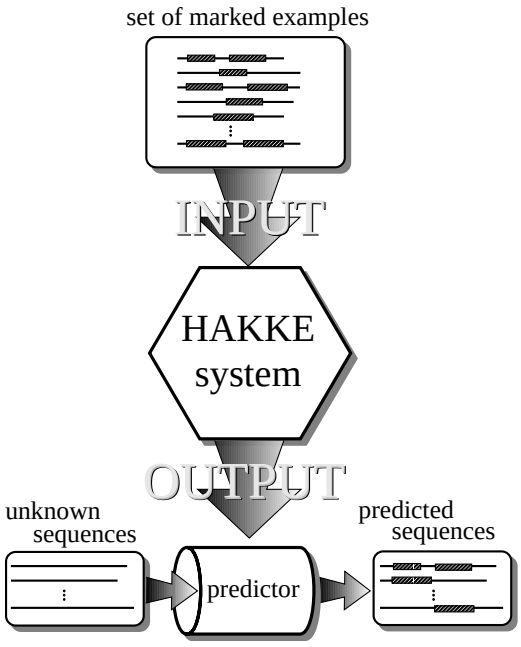

Figure 2: Overview of HAKKE

As input, HAKKE takes a set of marked examples for predicting a marking φ of an unknown sequence. After a number of weighted majority votes, the system produces a predictor which approximates the marking φ of the unknown sequence. HAKKE is a hybrid system which cooperates with a pool of prediction algorithms. Given a set S of marked examples, HAKKE can automatically generate candidate algorithms for the pool. In addition to these algorithms for the pool, users can supply *ad hoc* algorithms for the pool.

We assume that HAKKE has n prediction algorithms P_ℓ with weight w_ℓ for $1 \leq \ell \leq n$ in the pool. Given N marked examples $e_i = (s_i, m_i)$ for $1 \leq i \leq N$, HAKKE executes the following two stages for all $s_i[j]$ $(1 \leq i \leq N,\ 1 \leq j \leq |s_i|)$:

Stage 1: For each $1 \leq \ell \leq N$, P_ℓ takes s_i and an integer j as an input and predicts whether $s_i[j]$ is in some marked region of e_i or not. P_ℓ outputs one of the following three types of predictions: (1)" yes" (or 1), this means $s_i[j]$ is in some marked region, (2) "no" (or 0), $s_i[j]$ is not in any marked regions, and (3) "abstention" (or $*$).

Stage 2: The master algorithm computes the sum $score_1$ of all weights of the prediction algorithms whose prediction is 1. In the same way, the master algorithm computes the sum $score_0$ of all weights of the prediction algorithms whose prediction is 0. Then it compares $score_1$ and $score_0$ and makes a prediction p. According to the correctness of the prediction p, it determines the weight of P_ℓ.

A formal description of the algorithm is given in Figure 3. The master algorithm produces a predictor which consists of the prediction algorithms with the weights determined by repeating the above two stages for each marked example.

Input
  $S = \{(s_1, m_1),\ (s_2, m_2),\ \cdots,\ (s_N, m_N)\}$: a set of marked examples;
  $P_1, \cdots, P_n$: prediction algorithms;
Output
  $w_1, \cdots, w_n$: weights of prediction algorithms ($0 \leq w_\ell \leq 1$);
Algorithm *HAKKE*
begin
  forall $1 \leq \ell \leq n$ **do** $w_\ell := 1$;
  for $i := 1$ **to** N **do**
    for $j := 1$ **to** $|s_i|$ **do begin**
      $score_0 := score_1 := 0$;
      forall $k \in \{0, 1\}$ **do**
        forall ℓ such that the prediction of P_ℓ for $s_i[j]$ is k **do**
          $score_k := score_k + w_\ell$;
      if $score_1 > score_0$ **then** $p := 1$ **else** $p := 0$;
      if $p \neq m_i[j]$ **then begin**
        forall ℓ such that the prediction of P_ℓ for $s_i[j]$ is p **do**
          if $p = 1$ **then** $w_\ell := w_\ell \cdot \beta_1$ **else** $w_\ell := w_\ell \cdot \beta_0$;
        forall ℓ such that the prediction of P_ℓ for $s_i[j]$ is $*$ **do**
          $w_\ell := w_\ell \cdot \gamma$;
      end;
    end;
  output $w_1, \cdots, w_n$;
end.

Figure 3: Weighted majority algorithm with abstention

The accuracy of a predictor produced by HAKKE is represented by a 5-tuple (p_0,p_1,p_2,p_3,p_4), where p_0, called the *total accuracy*, means the ratio of correctly predicted positions to the whole sequences, p_1% of marked regions are recognized as positive by the predictor, p_2% of unmarked regions are recognized as negative, p_3% of predictions for marked regions is correct, and p_4% of predictions for unmarked regions is correct. We define the score of the predictor by $p_0 \cdot p_1 \cdot p_2 \cdot p_3 \cdot p_4$. By employing a local search technique, HAKKE changes the penalties β_0, β_1 and γ and finds weights for the prediction algorithms with which the resulting predictor attains a higher score.

5 Method and Experiments

This section reports the method and experiments performed by a prototype HAKKE system for α-helices and transmembrane domains.

5.1 Method

The current version of HAKKE is not equipped with any *ad hoc* algorithms yet. In order to generate a pool of prediction algorithms from a set S of marked examples, we applied BONSAI system [4]. First we prepared two sets POS_w and NEG_w of strings of length w ($w = 1,3,5,\ldots$) from marked examples $(s,m) \in S$ as follows: For each position j in s, the string $s_{j,w} = s[(j - \lfloor w/2 \rfloor)..(j + \lfloor w/2 \rfloor)]$ is in POS_w if $m[j] = 1$, and $s_{j,w}$ is in NEG_w if $m[j] = 0$. That is, a substring of s whose length is w is in POS_w if its center is in a marked region, and in NEG_w otherwise. BONSAI takes POS_w and NEG_w as inputs, and produces a pair (DT_w, I) of a decision tree over regular patterns and an alphabet indexing (Figure 4) which classifies POS_w and NEG_w approximately correctly.

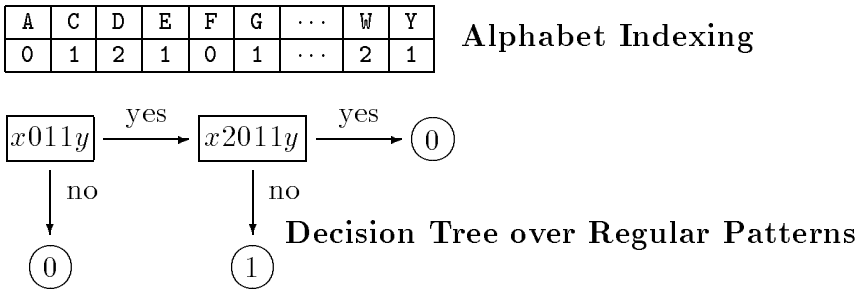

Figure 4: A decision tree over regular patterns and an alphabet indexing

We regard (DT_w, I) as a prediction algorithm for a pair (s,j) of a string s and an integer j with $1 \leq j \leq |s|$ as follows: Let $s_{j,w}$ be the substring of s of length w centered at j. First we transform $s_{j,w}$ into $I(s_{j,w})$ according to the alphabet indexing I. Then the decision tree DT_w determines the class name 0 ("not in any marked regions") or 1 ("in a marked region").

In order to introduce abstention, we constructed two kinds of decision trees DT_w^+ and DT_w^- from a decision tree DT_w in the following way: DT_w^+ is identical to DT_w except that all 0-labels of the leaves in DT_w are replaced with $*$. Similarly, DT_w^- is obtained from DT_w by replacing all 1-labels of the leaves with $*$. That is, DT_w^+ predicts only 1 or abstains, and DT_w^- predicts only 0 or abstains. In the experiments, we used these decision trees with alphabet indexings as prediction algorithms of the pool, and we set the penalty of abstention as $\gamma = 1$.

5.2 Experimental Results

5.2.1 Transmembrane domains

We used the PIR database [7], which contains the amino acid sequences with FEATURE field where transmembrane domains are indicated. In this experiment, we regarded the transmem-

brane domains as marked regions. We collected 356 amino acid sequences with transmembrane domains and we chose 36 sequences as marked examples that were used by HAKKE for generating a predictor.

The experimental results show that the prototype HAKKE system is sufficiently powerful to generate good predictors. Even without abstentions (shown in the last line of Table 1), the score 0.173 of the produced prediction algorithm is much higher than that of any prediction algorithms in the pool. Moreover, these experimental results show that the use of abstention makes the prediction ability higher: the score 0.207 using abstention is slightly higher than that without using abstention.

Table 1: Results on transmembrane domain sequences

	p_0 (%)	p_1 (%)	p_2 (%)	p_3 (%)	p_4 (%)	Score
DT_1	63.5	67.5	63.2	13.1	95.9	0.034
DT_3	75.8	71.5	76.1	19.8	97.0	0.079
DT_5	72.6	76.2	72.3	18.4	97.4	0.072
DT_7	76.7	71.7	77.1	20.5	97.1	0.084
DT_9	78.8	70.8	79.4	22.1	97.1	0.095
DT_{11}	82.2	65.8	83.6	24.8	96.7	0.109
DT_{13}	80.4	64.1	81.7	22.4	96.5	0.091
DT_{15}	81.8	62.0	83.4	23.6	96.4	0.096
DT_{17}	81.6	62.3	83.4	23.6	96.4	0.096
DT_{19}	82.4	56.6	84.5	23.1	95.9	0.087
DT_{21}	83.6	58.9	85.6	25.2	96.2	0.102
DT_{23}	80.4	59.1	82.1	21.4	96.1	0.080
generated predictor (with abstention)	91.6	54.6	94.6	45.6	96.2	0.207
generated predictor (without abstention)	87.9	65.5	89.8	34.6	96.9	0.173

5.2.2 α-helices

We used the PDB database [8] that contains the amino acid sequences associated with their three dimensional coordinates. By using the DSSP program [9] which translates three dimensional coordinates into the secondary structures, we marked the positions of α-helices in the amino acid sequences. We regarded the α-helices as the marked regions. We collected 4835 amino acid sequences with α-helices and 484 sequences were used as marked examples.

Table 2 shows the experimental results. From Table 2, we can observe that these experimental results on α-helices are similar to those on transmembrane domains.

Table 2: Results on α-helix prediction

	p_0 (%)	p_1 (%)	p_2 (%)	p_3 (%)	p_4 (%)	Score
DT_1	58.0	54.8	59.5	39.3	73.3	0.054
DT_3	60.0	57.0	61.5	41.5	74.9	0.065
DT_5	58.0	59.2	57.4	40.0	74.6	0.059
DT_7	58.7	58.4	58.9	40.5	74.7	0.061
DT_9	56.7	51.0	59.4	37.6	71.7	0.046
DT_{11}	57.7	50.5	61.1	38.4	72.0	0.049
DT_{13}	58.0	53.1	60.4	39.1	72.8	0.053
DT_{15}	55.7	48.1	59.4	36.2	70.5	0.041
DT_{17}	57.0	52.5	59.2	38.2	72.2	0.049
DT_{19}	57.0	52.9	59.0	38.2	72.3	0.049
DT_{21}	57.7	47.5	62.6	37.8	71.3	0.046
DT_{23}	57.7	50.8	61.0	38.4	72.1	0.050
generated predictor (with abstention)	65.3	44.5	75.3	46.4	73.9	0.075
generated predictor (without abstention)	61.6	56.3	64.2	43.0	75.4	0.072

6 Conclusion

We have described an overview of HAKKE and reported some experimental results obtained by a prototype system. The experimental results show that the idea of majority voting with abstention is useful in practice. We verified the power of the combination of plural algorithms, even if each algorithm is incomplete. The extended majority voting algorithm WM* achieves higher prediction accuracy than any algorithm in the pool. It also deserves to mention that the predictor produced by the system is easy to analyze. The algorithms with larger weights in the pool are more important in the predictor. Unlike the neural network approach, the knowledge mapped on the predictor can be fed back to the domain experts and may lead to discoveries.

Since the prototype system uses only automatically generated prediction algorithms and is not equipped with any *ad hoc* algorithms, the reported prediction accuracy is not very good. The accuracy can be improved by searching better algorithms for the pool or employing suitable *ad hoc* algorithms on the specific problem domains.

In future, we will tune up the system especially for "gene finding".

Acknowledgment

This work is supported in part by a Grant-in-Aid for Scientific Research on Priority Areas "Genome Science" from The Ministry of Education, Science, Sports and Culture in Japan. S. Matsumoto is a Research Fellow of JSPS and partly partly supported by Grants-in-Aid for JSPS research fellows.

References

[1] P. Y. Chou and G. D. Fasman, "Prediction of the Secondary Structure of Proteins from Their Amino Acid Sequence," *Advances in Enzymology*, Vol. 47, pp. 45–147, 1978.

[2] J. Kyte and R. F. Doolittle, "A Simple Method for Displaying the Hydropathic Character of a Protein," *J. Mol. Biol.*, Vol. 157, pp. 105–132, 1982.

[3] N. Littlestone and M. K. Warmuth, "The Weighted Majority Algorithm," *Information and Computation*, Vol. 108, pp. 212–261, 1994.

[4] S. Shimozono, A. Shinohara, T. Shinohara, S. Miyano, S. Kuhara and S. Arikawa, "Knowledge Acquisition from Amino Acid Sequences by Machine Learning System BONSAI," *Trans. Information Processing Society of Japan*, Vol. 35, pp. 2009–2018, 1994.

[5] E. C. Uberbacher and R. J. Mural, "Locating Protein-Coding Regions in Human DNA Sequences by a Multiple Sensor-Neural Network Approach," *Proc. Natl. Acad. Sci USA*, Vol. 88, pp. 11261–11265, 1991.

[6] Y. Xu, J. R. Einstein, R. J. Mural, M. Shah and E. C. Uberbacher, "An Improved System for Exon Recognition and Gene Modeling in Human DNA Sequences," *Proc. the Second Internatinal Conference on Intelligent Systems for Molecular Biology*, AAAI Press, pp. 376–383, 1994.

[7] National Biomedical Research Foundation, "Protein Identification Resource," 1994.

[8] F. C. Bernstein, T. F. Koetzle, G. J. Williams, E. E. Meyer Jr, M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi and M. Tasumi, "The Protein Data Bank: a Computer-Based Archival File for Macromolecular Structures," *J. Mol. Biol.*, Vol. 112, pp. 535–542, 1977.

[9] W. Kabsch and C. Sander, "Dictionary of Protein Secondary Structure: Pattern Recognition of Hydrogen-Bonded and Geometrical Features," *Biopolymers*, Vol. 22, pp. 2577–2637, 1983.