Classifying Transport Proteins Using Profile Hidden Markov Models and Specificity Determining Sites

Qing Ye

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science (MCompSc) at

Concordia University

Montréal, Québec, Canada

April 2019

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Qing Ye**

Entitled: **Classifying Transport Proteins Using Profile Hidden Markov Models and Specificity Determining Sites**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science (MCompSc)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. T.-H. Chen

_____ Examiner
Dr. T. Glatard

_____ Examiner
Dr. A. Krzyzak

_____ Supervisor
Dr. G. Butler

Approved by _____
Martin D. Pugh, Chair
Department of Computer Science and Software Engineering

_____ 2019

_____
Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

Classifying Transport Proteins Using Profile Hidden Markov Models and Specificity Determining Sites

Qing Ye

This thesis develops methods to classifiy the substrates transported across a membrane by a given transmembrane protein. Our methods use tools that predict specificity determining sites (SDS) after computing a multiple sequence alignment (MSA), and then building a profile Hidden Markov Model (HMM) using HMMER. In bioinformatics, HMMER is a set of widely used applications for sequence analysis based on profile HMM. Specificity determining sites (SDS) are the key positions in a protein sequence that play a crucial role in functional variation within the protein family during the course of evolution.

We have established a classification pipeline which integrated the steps of data processing, model building and model evaluation. The pipeline contains similarity search, multiple sequence alignment, specificity determining site prediction and construction of a profile Hidden Markov Model.

We did comprehensive testing and analysis of different combinations of MSA and SDS tools in our pipeline. The best performing combination was MUSCLE with Xdet, and the performance analysis showed that the overall average Matthews Correlation Coefficient (MCC) across the seven substrate classes of the dataset was 0.71, which outperforms the state-of-the-art.

Acknowledgments

I am extremely thankful to my supervisor Dr. Gregory Butler for his expertise, time, patience, continuous support, warm-hearted and encouragement extended to me. My deepest gratitude goes to my parents for their unquestioning and unconditional love. I give thanks to my father for his giving me continuous economic support throughout my research period, mother for her always trust and encouragement. My sincere thanks to my lab mates: Faizah Aplop, Lin Cheng, Stuart Thiel, Stephanie Kamgnia, Munira AlBalla, Akhil Jobby and Shiva Shamloo for being a great source of friendships and providing me with lots of academic, moral, personal and professional support.

Contents

List of Figures

List of Tables

Glossary

AAC Amino acid composition: the frequency of each amino acid in a protein

Accuracy The ability of the classifier to find the number of correct decision (both positive and negative) among the total number of cases examined

Alignment The process, or its result, of matching sequences to maximize an objective function

Amino acid One of the 20 chemical building blocks that form a polypeptide chain of a protein

ATP Adenosine triphosphate

BLAST Basic Local Alignment Search Tool: a heuristic algorithm for pairwise sequence alignment

`blastp` BLAST program to search a proten sequence as a query against a database of protein sequences

Blast+ Software package from NCBI which is latest version of implementation of BLAST

BLOSUM Blocks substitution matrix

Clustal Family of algorithms for multiple sequence alignment

ClustalW A commonly used progressive multiple sequence alignment program

Clustal Omega The latest multiple sequence alignment program from the Clustal family

HMM Hidden Markov Model

HMMER Commonly used software package for sequence analysis based on Profile Hidden Markov Model

IMP Integral membrane proteins are permanently attached to a membrane

JDet Multiplatform software for the interactive calculation and visualization of function-related conservation patterns in multiple sequence alignments and structures

MAFFT Multiple alignment program for amino acid or nucleotide sequences

MCC Matthews correlation coefficient, which is a single measure taking into account true and false positives and negatives

Motif Conserved element of a protein sequence alignment that usually correlates with a particular function

MUSCLE Multiple sequence comparison by log-expectation: software for MSA

Ortholog Orthologs are genes in different species that evolved from a common ancestral gene by a speciation event forming two separate species

PAM Point accepted mutation matrix

Pfam Collection of protein families represented by multiple sequence alignments and hidden Markov models (HMMs)

Profile Sequence profile is usually derived from multiple alignments of sequences with a known relationship, and represented as a PSSM or HMM

Protein Macromolecule that consists of a sequence of amino acids

PSSM Position Specific Scoring Matrix

Sensitivity The ability of the classifier to detect the positives that are correctly identified as such

SDS Specificity determining site

SimGroup A program for protein specificity determining position predictions

SP Sum-of-pairs function

SPEER SERVER A tool for prediction of protein specificity determining sites

Specificity The ability of the classifier to detect the negatives that are correctly identified as such

Swiss-Prot A high quality annotated and non-redundant protein sequence database

T-Coffee Tree based consistency objective function for aligment evaluation

TC Transporter classification scheme of IUBMB

TCDB Transporter classification database www.tcdb.org

TM-Coffee One part of the T-Coffee package that is designed specifically to align transmembrane proteins

TMS Transmembrane segment

Transmembrane protein Protein that spans the membrane

Transport The directed movement of a molecule into, out of, or within a cell, or between cells

TransportDB Transporter database primarily for prokaryotes

Transporter Protein carrying out transport

Xdet Binary distribution of functional residue detection programs of Jdet, used outside the JDet environment, directly from the command line, for massive/batch runs

Chapter 1

Introduction

Our work develops and evaluates classifiers for the class of substrates that are transported across a membrane by a transmembrane transport protein. The substrate class is typically categorized as amino acid, sugar, ion (electron, anion, cation), protein, phosphate, etc. Each represents a class of specific chemical compounds.

Our classifiers are constructed as Hidden Markov Model (HMM) using a pipeline of multiple sequence alignment (MSA), prediction of specificity determining sites (SDS), and the HMMER tool `hmmbuild`.

In this chapter, Section 1.1 provides a brief summary on the basic biological knowledge of membrane and transmembrane proteins; Section 1.2 presents the state of the art; Section 1.3 shows the contribution of this thesis; Section 1.4 is the layout of this thesis.

1.1 Biological Background

The cell's plasma membrane separates the inside and outside of a cell, as Figure 1.1 shows. For eukaryotes, membranes also define the intracellular organelles [32]. All of these biomembranes are a phospholipid bilayer that contain embedded proteins, as shown in Figure 1.1.

Membrane proteins are important proteins for organisms. In the yeast *Saccharomyces cerevisiae*, for example, about one-third of its genes encode membrane proteins. In multicellular organisms the ratio is even higher. Membrane proteins are responsible for vital functions, such as,

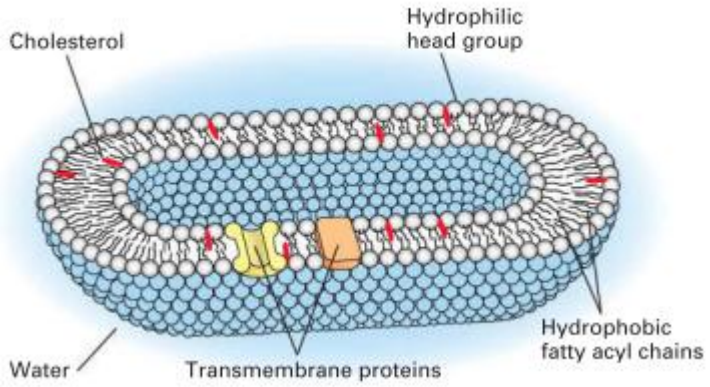

Figure 1.1: The cell surrounded by plasma membrane
The cell membrane separates the inside and outside of the cell, and in eukaryotes, there is also a membrane for organelles separating the inside and outside of the organnelle [32].

molecule and ion transport between inside and out of cells and organelles, and adhesion and interactions between cells [32].

There are three classes of membrane proteins: integral membrane proteins (IMP), lipid-anchored membrane proteins and peripheral membrane proteins. In this thesis, we concentrate on the integral membrane proteins, which are also called transmembrane proteins. In particular, we are concerned with the transmembrane transport proteins.

Membrane proteins play an important role in the living cells for energy production, regulation and metabolism [29]. Nowadays around half of the new drugs target membrane proteins [9], especially for their important function of moving signals and compounds through the membrane.

However, the structure and function information available for membrane proteins is very limited. The main reason is that membrane proteins have partially hydrophobic surfaces and can only be extracted from the cell membrane with detergents: the protein's flexibility and lack of stability make the extraction very difficult. Furthermore, the crystallization, expression, and structure determination of membrane proteins is also difficult [12]. That is why, in 2015, the Protein Data Bank (PDB) contains only 1% membrane proteins amongst its more than 112,000 protein structures. So we rely on computational techiniques to predict the function and structure of a membrane protein based on its similarity with other well-characterized proteins [32].

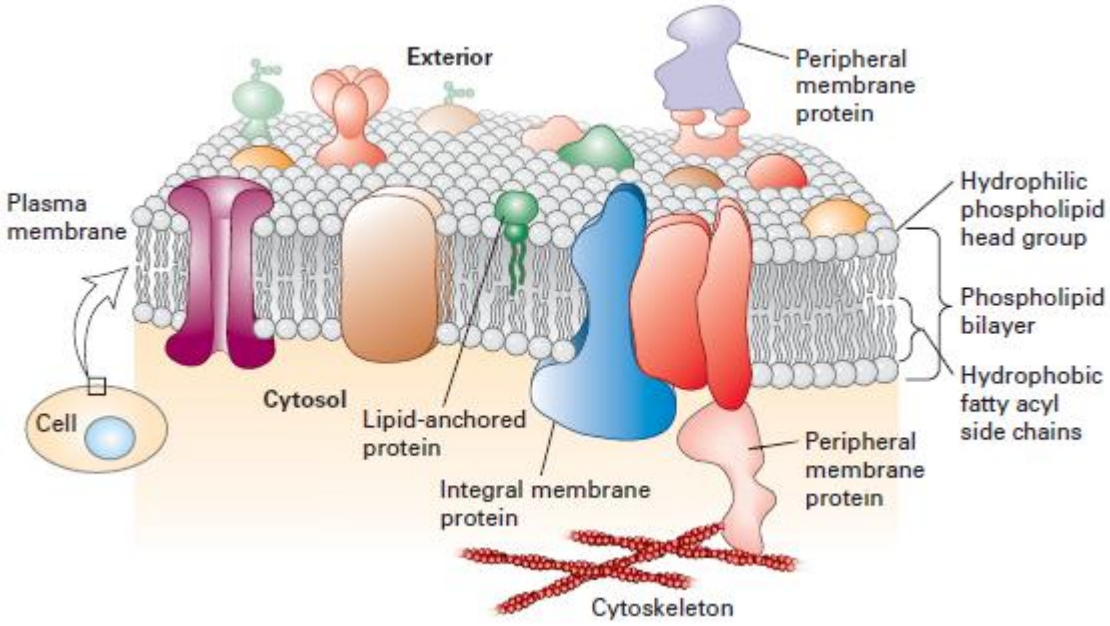

Figure 1.2: Biomembrane and Embedded Proteins

Membranes are composed of a phospholipid bilayer that can accommodate embedded transmembrane proteins and form attachment or adhesion points for other membrane proteins [32].

1.2 Transmembrane Protein Classification Methods

Previous work has applied machine-learning methods for the classification of transport proteins [4]. The Zhao Lab has developed TransportTP, a two phase algorithm that combines homology and machine learning to predict Transporter Classification TC family of one or more proteins. The first phase uses homology methods to predict the TC family based on sequence similarity to the classified proteins in the TCDB database. The second phase integrates different machine learning methods on a variety of features to refine the initial predictions [31]. Gromiha and Yabuki [23] have utilized different machine learning algorithms, such as, Bayes rule, Logistic regression, Neural network, Support vector machine, and Decision tree to discriminate different classes of transporter proteins. They also report a k-nearest neighbor method using amino acid composition to discriminate transporters and non-transporters. Furthermoe,, amino acid composition [**?**] is used to measured the similarity of membrane transporters from *Arabidopsis thaliana*, and a more accurate result was obtained by splitting the amino acid composition of TMS and non-TMS regions [43]. The state-of-the-art is TrSSP [33] uses a Support Vector Machine (SVM) with input from a profile PSSM and the AAIndex physico-chemical composition.

Some methods for prediction are based on sequence similarity. The principle is that proteins of high sequence similarity are typically related by evolution — so-called **homologous** sequences — and thus belong to the same family. However homologous sequences do not always share significant sequence similarity, and homologous proteins often have different functions [10]. The specificity of substrate for a transport protein can depend on a small number of residues at specific sites in the protein [4], and the previous methods all ignore the specificity determining sites (SDS). So this thesis incorporates SDS to aid the prediction.

1.3 Research Contributions

The contributions of this thesis are as follows:

(1) This work is the first to apply Hidden Markov Models to classify the substrate class of a transmembrane transport protein, and the first to combine the specificity determining sites

with profile Hidden Markov Model to the classification for the substrate class transported across a membrane by a given transmembrane transporter protein.

(2) We have established a classification pipeline which integrated the steps of data pre-processing, model building and model evaluation. The pipeline contains similarity search, multiple sequence alignment, specificity determining site prediction and construction of a Hidden Markov Model.

(3) We discuss and analyze the impact of the using different MSA tools, SDS tools and parameters on the classification of the substrate class. We find a combination which outperforms the state-of-the-art.

1.4 Organization of Thesis

This thesis is organized as follows: Chapter 2 introduces the background necessary for understanding the work in this thesis. Chapter 3 presents the details of our pipeline implementation. Section 3.1 presents the material, that is, datasets and databases, used in the project; Section 3.2 specifically describes the approach and its steps. Section 3.3 and Section 3.4 respectively shows the experimental results and discussion on our work. Chapter 4 concludes the thesis. Section 4.2 presents the limitations in our work, and section 4.3 suggests some directions worthy of further investigation.

Chapter 2

Background

2.1 Membrane Proteins

2.1.1 Protein: A Sequence of Amino Acids

As one of the most important chemical building blocks of cells, the amino acid has a central alpha carbon atom, and four chemical groups bonded to it: an amino $-NH_2$, a carbboxyl or carboxylic acid $-COOH$, a hydrogen atom H, and a side chain. The distinctive property of amino acids are determined by their side chains. Along with other characteristics: the size, shape, charge, hydrophobicity, the amino acids are divided into several categories, shown in Figure 2.1.

Proteins are polymers of amino acids. As Figure 2.2 shows, the function of the protein is determined by its structure. In detail, the protein function is derived from its 3-Dimensional structure, and the 3-Dimensional structure is determined by both its amino acid sequence and intramolecular noncovalent interactions[32]. So the linear sequence of amino acids is called the protein primary structure, which is shown in Figure 2.1.1. A single letter code can be used to represent the 20 naturally occurring amino acids. The 20 amino acids detailed in Figure 2.1.1. There is an example for sequence of amino acids in Table 2.1, which is the protein sequence of human beta globin taken from Swiss-Prot database.

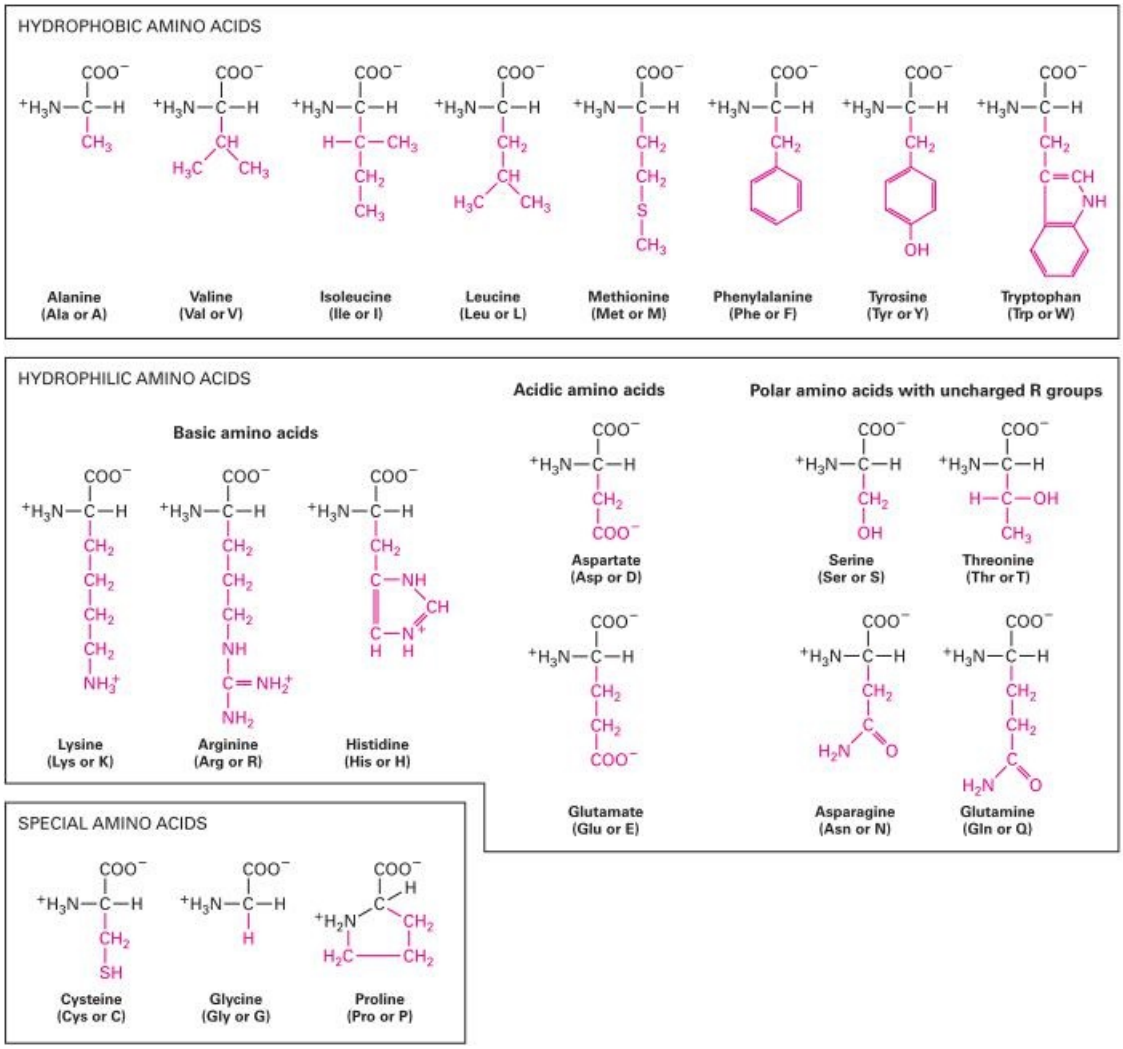

Figure 2.1: The 20 amino acids and side chains (in red)

The chemical structure of the 20 amino acids showing side chains [32].

>sp |*P*68871| HBB_HUMAN Hemoglobin subunit beta OS=Homo sapiens OX=9606 GN=HBB PE=1 SV=2
MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPK
VKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFGK
EFTPPVQAAYQKVVAGVANALAHKYH

Table 2.1: An example of protein sequence in fasta format.

The fasta format is a text-based format for representing either nucleotide sequences or peptide sequences, in which base pairs or amino acids are represented using single-letter codes. A sequence in fasta format begins with a single-line description, followed by lines of sequence data. The description line is distinguished from the sequence data by a greater-than (">") symbol in the first column. It is recommended that all lines of text be shorter than 80 characters in length.

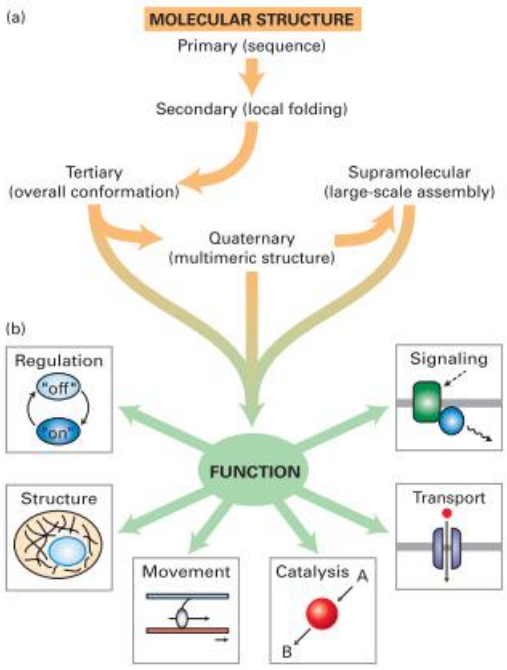

Figure 2.2: Protein structure and function

An illustration of the primary, secondary, tertiary and quaternary structures of a protein and protein complexes, together with larger supermolecular combinations of proteins (a). The structure determines function. The figure shows the various functional roles (b). [32]

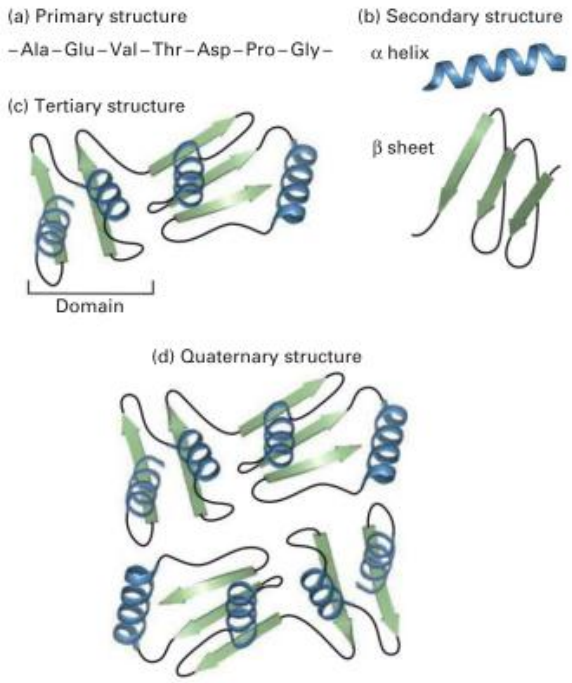

Figure 2.3: Four levels of protein hierarchy

An illustration of the primary, secondary, tertiary and quaternary structures of a protein and protein complexes. [32]

2.1.2 Integral Membrane Proteins

As discussed in Section 1.1, the cell's plasma membrane separates the inside and outside. The integral membrane proteins transport specific ions, sugars, amino acids, and vitamins to cross the impermeable phospholipid bilayer into the cell and export metabolic products out. They also process a similar exchange for intracellular organelles.

The integral membrane proteins contain three domains: the membrane-spanning segments usually consisting of hydrophobic amino acids whose side chains interact with the the phospholipid bilayer; the cytosolic and exoplasmic domains have hydrophilic exterior surfaces that interact with the aqueous environment of each side of the membrane. Until now, all the membrane-spanning domains are built by one or more α-helices or of multiple β-strands. The integral membrane proteins create a protein-lined pathway across the membrane, to transport hydrophilic substances through the membrane without touching its hydrophobic interior.

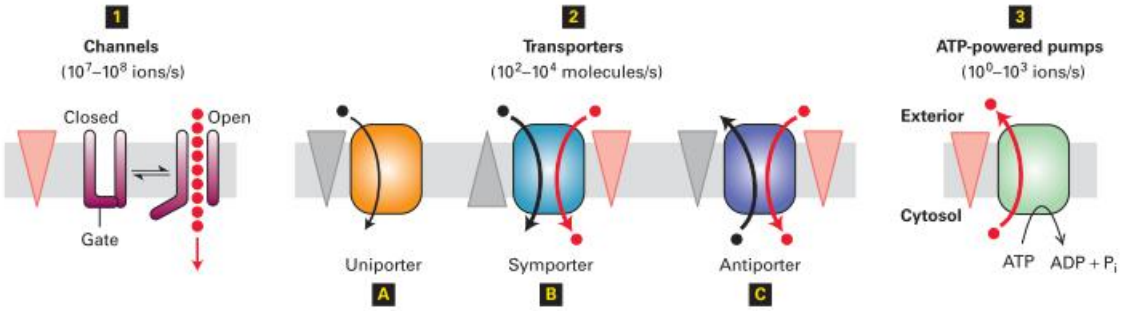

Figure 2.4: Three main types of integral membrane proteins

The three main types of integral membrane proteins: channels, transporters, and pumps. Channels transport water, ions, or hydrophilic small molecules by their concentration or electric potential gradients. Transporters move a wide variety of molecules, and have three types: uniporters transport a molecule down its concentration gradient; symporters move a molecule against its concentration gradient by allowing a different molecule down its concentration gradient, that is in the same direction; antiporter move a molecule against its concentration gradient by allowing a different molecule down its concentration gradient, that is in the opposite direction; ATP-powered pumps use ATP to move ions or small molecules against a gradient or an electric potential. [32]

There are three types of main types of integral membrane proteins, shown in Figure 2.4. Channels transport water, ions, or hydrophilic small molecules by their concentration or electric potential gradients. Transporters move a wide variety of ions and molecules across cellular membranes, but at a much slower rate: uniporters transport only type of molecule down its concentration gradient;

antiporters and symporters move one type of ion or molecule against its concentration gradient by allowing one or more different ions down its concentration gradient. ATP-powered pumps are ATP powered to move ions or small molecules against a chemical concentration gradient, an electric potential, or both. Figure 2.5 shows the structure and function of one ATP-powered pump.

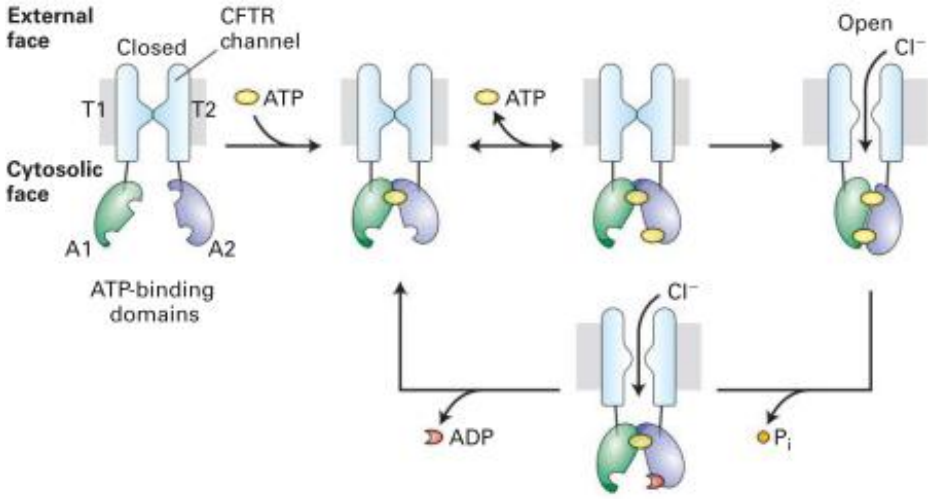

Figure 2.5: Structure and function of the cystic fibrosis transmembrane regulator
An illustration of the mechanism by which an ATP-powered pump is opened and closed to control the transport of a molecule. [32]

2.2 Multiple Sequence Alignment

2.2.1 Protein: A Sequence of Amino Acids

The primary structure of a protein is the linear sequence of amino acids [42]. A single letter code can be used to represent each of the 20 naturally occurring amino acids, detailed in Table 2.2.

2.2.2 Mathematical Definition of Multiple Sequence Alignment

A protein sequence s of length l is a string of l characters derived from the alphabet

$$\mathcal{A} = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$$

Amino Acid	Letter	Amino Acid	Letter
Alanine	A	Arginine	R
Asparagine	N	Aspartic acid	D
Cysteine	C	Glutamic acid	E
Glutamine	Q	Glycine	G
Histidine	H	Isoleucine	I
Leucine	L	Lysine	K
Methionine	M	Phenylalanine	F
Proline	P	Serine	S
Threonine	T	Tryptophan	W
Tyrosine	Y	Valine	V

Table 2.2: The 20 naturally occurring amino acid notations.

. Given a sequence S containing N sequences: $S = \{S_1, S_2, ..., S_N\}$, $N \geqslant 2$, $S_i = S_{i1}, S_{i2}, ..., S_{il_i}$, for $1 \leqslant i \leqslant N$, and $S_{ij} \in \mathcal{A}$, for $1 \leqslant j \leqslant l_i$, where l_i is the length of the ith sequence, then a **multiple sequence alignment** can be defined as a matrix $A = (a_{ij})$, $1 \leqslant i \leqslant l$, $max(l_i) \leqslant l \leqslant \sum_{i=1}^{N} l_i$, shown as Figure 2.6. The matrix must meet the following three conditions:

(1) $a_{ij} \in \mathcal{A} \cup \{-\}$, where "–" stands for gap.

(2) After the "–" is removed from the ith line in the matrix, the string S_i is obtained.

(3) The matrix does not contain columns whose characters are all gaps.

```
P69905 (HBB_HUMAN) MV-LSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-DLSH-----GS 53
P68871 (HBB_HUMAN) MVHLTPEEKSAVTALWGKV--NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGN 58
P02144 (MYG_HUMAN) -MGLSDGEWQLVLNVWGKVEADIPGHGQEVLIRLFKGHPETLEKFDKFKHLKSEDEMKAS 59
                    : *:  :   *   ****         * * *::   * *   *  *  *

P69905 (HBB_HUMAN) AQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAH 113
P68871 (HBB_HUMAN) PKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHH 118
P02144 (MYG_HUMAN) EDLKKHGATVLTALGGILKKKGHHEAEIKPLAQSHATKHKIPVKYLEFISECIIQVLQSK 119
                     :* **  *  *:   :          :  *:: *  *  :    : ::   ::  *  :

P69905 (HBB_HUMAN) LPAEFTPAVHASLDKFLASVSTVLTSKYR------ 142
P68871 (HBB_HUMAN) FGKEFTPPVQAAYQKVVAGVANALAHKYH------ 147
P02144 (MYG_HUMAN) HPGDFGADAQGAMNKALELFRKDMASNYKELGFQG 154
                      :*    : : :*        :: :*:
```

Figure 2.6: Multiple Sequence Alignment

Three polypeptide chains from the globin family aligned by the ClustalW program and shown together with their UniProt accession code (and short name in the database).

2.2.3 Scoring Multiple Sequence Alignments

The purpose of scoring for multiple sequence alignment is to provide a measure of the similarity between multiple sequences. The **objective function** is a metric used to check the quality of a multiple sequence alignment, and all multiple sequence alignment methods rely on an objective function to illustrate the quality of the alignment, which reflects the accuracy and effectiveness of the method. The most common objective function used is the sum-of-pairs function (SP). The SP function needs to set two important parameters: substitution matrix M and gap penalties (including open gap penalties and extension gap penalties).

The calculation formula of the Sum-of-Pairs function can be expressed by $score$ equal to the sum of the residue scores and the penalty scores. The $score$ is defined as a positive number, where the higher the $score$, the better the alignment. The total reside score is defined by the formula

$$\sum_{h=1}^{L} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} cost(S_{ih}, S_{jh})$$

where

$$cost(S_{ih}, S_{jh}) = \begin{cases} M(S_{ih}, S_{jh}) & \text{If both are the same residue (match);} \\ M(S_{ih}, S_{jh}) & \text{If both are residues, but different (non-match);} \\ 0 & \text{If either is a gap.} \\ & \text{calculated).} \end{cases}$$

where L is the length of the alignment; m is number of sequences to participate in the alignment; S_{ih} is h-th residue of the i-th sequence. The scores for matching and mismatching are usually given by the substitution scoring matrix.

For protein sequences, the **substitution scoring matrix** is primarily used to record the similarity of two corresponding residues in sequence alignment. Once the matrix is defined, the alignment program can take advantage of this matrix and try to rank similar residues together to achieve the best alignment. Commonly used substitution scoring matrices are point accepted mutation(PAM) and blocks substitution matrix (BLOSUM) [25]. The most commonly used are the PAM250 matrix and the BLOSUM-62 matrix. The PAM250 similarity score matrix is equivalent to a 20% residue

match between the two sequences.

In the alignment process, a **gap** is introduced to indicate insertion and deletion. In general, because multiple sequences participating in the alignment are not identical, to align them, you need to insert gaps.

In multiple sequence alignment, if you choose to insert a large number of gaps, then any two random unrelated sequences can be aligned, but such alignment results may have no biological significance, so the gap must be limited to some extent. The number is used to produce a biologically meaningful alignment result, using a scoring strategy: the matched residues get a positive score, while the gaps get a negative score or a penalty.

The **gap penalty** consists of two parts: the **gap open penalty** and the **gap extend penalty**. In general, the gap open penalty is higher than the gap extend penalty. The gap open penalty refers to the first gap inserted in a sequence when the sequence is aligned. The gap extend penalty means that after introducing one or more gaps, the continuation of the next consecutive gap is continued.

The scores of the gap open penalty and the gap extend penalty directly affect the results of the sequence alignment, as shown in Table 2.3 [25].

Gap Open Penalty	Gap Extend Penalty	Effect on Results of the Sequence Alignment
Large	Large	Very few insertions and deletions, suitable for sequence alignments withhigher similarity
Large	Small	Small amount of large gaps inserted
Small	Large	Large number of small block inserts, suitable for sequence alignment with lower similarity

Table 2.3: The effect of the gap open penalty and the gap extend penalty.

2.2.4 Algorithm for Multiple Sequence Alignment

Heuristic, approximate algorithms are used for multiple sequence alignment due to the fact that an exact algorithm based on multi-dimensional dynamic programming is simple too complex and computationally expensive.

There are several main types of heuristic algorithms: progressive algorithm, iterative algorithm and consistency-based algorithm. The idea of the **progressive algorithm** is to first construct a

distance matrix by pairwise alignment of the sequences to show the relationship between two sequences; then calculate a phylogenetic guide tree according to the distance matrix, and weight the closely related sequences. Then, beginning with the two closest sequences, the adjacent sequences are gradually introduced and the alignment is continuously re-established until all sequences have been added. Due to its simple and fast characteristics, the progressive algorithm is one of the most commonly used methods in multiple sequence alignment. However, since the nature of the progressive algorithm is a greedy algorithm, the results of initial steps of the algorithm cannot be changed during the processs. This results in a "local minimization" problem. The representative progressive algorithms include ClustalW and Clustal Omega. An **iterative alignment algorithm** is an effective strategy. It iteratively refines an alignment until the alignment results are no longer improved. The disadvantage of this type of algorithm is that it does not provide guarantees for obtaining optimized results, and the speed cannot be compared with the progressive algorithm. The advantage is that the objective function and the optimization process are conceptually separated, and it has the characteristics of robustness and insensitivity to the number of sequences. The representative progressive algorithms include MAFFT and MUSCLE. The idea behind **consistency-based methods** is that, for sequences x, y and z, if residue x_i aligns with residue y_j and y_j aligns with z_k, then x_i aligns with z_k. The consistency of each pair of residues to the residue pairs from all other alignments is examined and weighted in such a way as to reflect the extent to which these residues are consistent with other residues. The consistency-based approach often generates final multiple sequence alignments that are more accurate than those achieved by progressive alignments, based on benchmarking studies [38]. The representative consistency-based algorithms include T-Coffee.

2.3 Specificity Determining Sites

This section discusses the important processing stage to discover the specificity determining sites (SDS), which are the key positions in the sequences of a protein for functional divergence [14].

2.3.1 Functional Divergence

After gene duplication events, the chromosome contains one gene copy which maintains the original function, and multiple copies of the gene which accumulate amino acid changes that can lead to functional divergence [24]. As a result, genes are represented as paralogs in the genome with related but distinct functions. However most amino acid changes are not related to functional divergence but represent neutral evolution [28]. So it is crucial to distinguish between these two possibilities. Similarity search and multiple sequence alignment are not enough to solve this problem.

2.3.2 Types of Sites

It is generally believed that the more important in function of amino acid site is, the less possible for its evolutionary change of amino acid, because the variance in this kind of site will highly reduce the possibility of its host to survive and reproduce [28]. When the function of the protein changes due to the change in amino acid, then the site is an example of a specificity determining site for the protein subfamilies, based on a *general* function for the family. The SDS have special conservation or evolutionary role within a protein family [14], comparing to the neutral, non-function-related mutations in the majority of amino acid sites.

Three types of functional divergences are distinguished, mainly based on the pattern of evolution giving rise to **evolutionary rate-baes approachea** to SDS prediction. **Type I** functional divergence is the result of significant rate difference at a given site between two subgroups of a protein family indicating that the function constraints at this position are different in the two groups. Type I functional divergence after gene duplication is highly correlated with the change in evolutionary rate, which is analogous to a fundamental rule in molecular evolution: functional importance is highly correlated with evolutionary conservation [28]. The site of type I functional divergence is conserved for one subfamily and is variable in another, which is shown in Table 2.4. The **type II** specificity determining sites may show a subfamily-specific conservation pattern, which is conserved in all subfamilies but using different amino acids in different subfamilies. This type II divergence is a consequence of the rate change where selection causes similar levels of conservation of di! fferent

amino acid types for different protein subfamilies. Recent studies have also identified a third type of site (**type MC**; marginally conserved) involved in subfamily specificity determination where no apparent conservation of amino acids is observed within any of the subfamilies [13]. An example is given in Figure 2.7 [13].

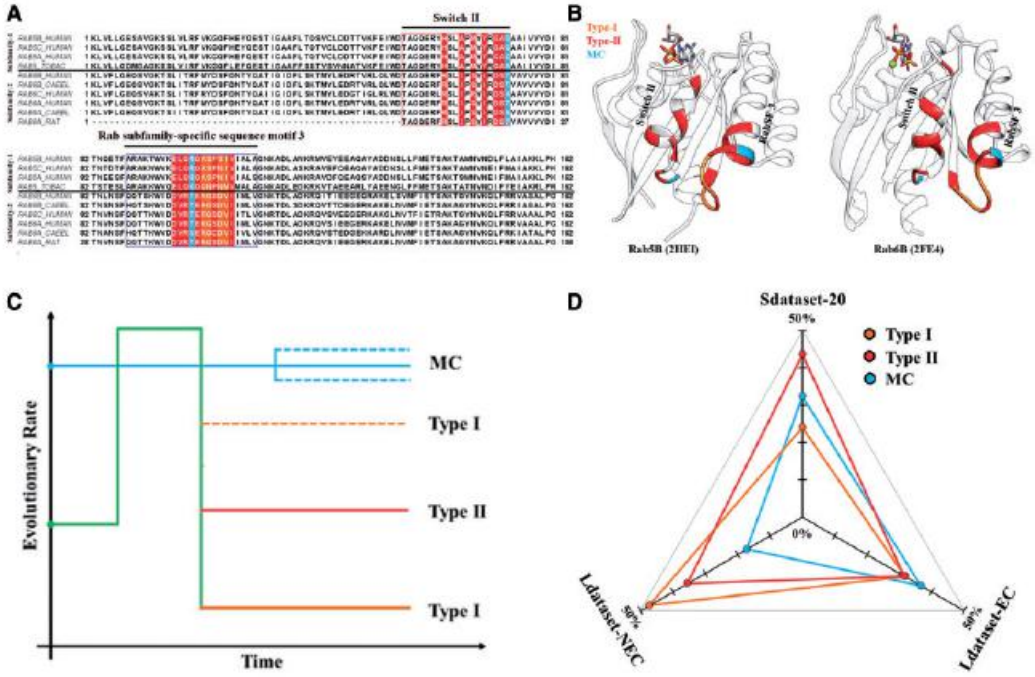

Figure 2.7: Types of Sites

In this figure, the Rab protein family has two subfamilies and all three types of SDS. The corresponding amino acid sequences (A), 3D structure (B) and evolutionary rates (C) are shown. the multiple sequence alignment (A) shows that type I and type II sites have regular conservation pattern, but type MC sites are not regular and are hard to recognize. The evolutionary rate of change graph (C) shows that significant changes have happened for type I and type II sites, but much smaller changes for type MC sites. [14]

Chakrabarti et al. [13] indicate that almost half of the SDS belong to MC type. So using only conservation/evolution information to distinguish SDS from a strong background signals is extremely difficult. Different algorithms have been developed during the past decades, combining at least some of the signals from MSA, phylogenetic tree, amino acid physico-chemical properties and the protein's 3D structure [14].

Subfamilies	Sequences	Type I		Type II		Type MC	
Subfamily 1	Sequence 1	A	S	H	A	C	R
	Sequence 2	A	S	H	A	C	R
	Sequence 3	A	S	H	A	C	R
	Sequence 4	G	N	H	A	A	K
Subfamily 2	Sequence 5	R	G	R	I	T	T
	Sequence 6	R	G	R	I	T	T
	Sequence 7	R	G	R	I	A	T
	Sequence 8	R	G	R	I	T	T
	Sequence 9	R	G	R	I	S	N
	Sequence 10	R	G	R	I	T	T

Table 2.4: Three types of specificity determining sites.
For type I functional divergence, the site is conserved for one subfamily and is variable in another. The type II divergence causes similar levels of conservation of different amino acid types for different protein subfamilies. The type MC sites are marginally conserved and no apparent conservation of amino acids is observed within any of the subfamilies. [13]

2.3.3 Prediction Methods

Diffferent computational algorithms have been developed during the past years. During them, there three basic approaches: evolutionary rate-based, entropy-based, and amino acid physicochemical properties-based. The approach based on evolution was discussed above. The details of the other two approaches will be discussed below.

Here we introduce **entropy-based approaches** as an example [1, 48]. Given an MSA, the amino acid distribution p_c of each column c is computed by:

$$p_c(x) = \frac{n_x + AK_x}{N + A} \tag{1}$$

where n_x is the number of occurrences of amino acid x in column c, N is the total number of amino acids in the column, A is weighting factor, and K_x is the frequency of the amino acid x derived from the Swiss-Prot protein sequence database[5].

The relative entropy $p_c(x)$ and $q_c(x)$in one position between two subfamilies p and q is computed as:

$$REp = \sum_x p_c(x) \log \frac{p_c(x)}{q_c(x)} \tag{2}$$

$$REq = \sum_x q_c(x) \log \frac{q_c(x)}{p_c(x)} \tag{3}$$

and the cumulative relative entropy (CRE)is

$$CRE = REp + REq \tag{4}$$

The CRE metric for each position was then converted into a Z-score, computed as

$$Z = \frac{CRE - \mu}{\sigma} \tag{5}$$

where μ is the arithmetic mean and σ is the standard deviation of the CRE values observed in all positions of the subfamily comparison.

The Z-score is normalization for the similarity between two distributions of amino acids. The larger the Z-score value, the more the amino acid distributions in both subfamilies differ from one another. By using the Z-score normalization we can treat different families in the same scale. The SDS site will be detected by the amino acid sites whose Z-score value exceed a threshold 0.5.

There are several ways to capture amino acid properties as used by the **physico-chemical properties-based approaches**. Dubchak et al [17] divided 20 amino acids into three groups according to the values in four attributes, which is shown in Figure 2.8. For each attribute, every amino acid is assigned by the value 1, 2, or 3 according to one of the three groups to which it belongs. Based on the values, they calculate the features of composition (C), transition (T) and distribution (D) for the given amino acid sequence. The C-features, $C_k = \frac{n_k}{N}$, $k = 1, 2, 3$, represent the weight of each group in any attribute of amino acids in the total sequence. The T-features, $T_{k,t} = \frac{n_{kt}+n_{tk}}{N-1}$, $tk = 12, 23, 13$, represent the percent frequency with which (1) a polar residue is followed by a neutral residue, or a neutral residue by a polar residue; (2) a polar residue is followed

Property	Group 1	Group 2	Group 3
Hyrophobicity15	Polar	Neutral	Hydrophobic
	R, K, E, D, Q, N	G, A, S, T, P, H, Y	C, V, L, I, M, F, W
Normalized van der Waals volume16	0–2.78	2.95–4.0	4.43–8.08
	G, A, S, C, T, P, D	N, V, E, Q, I, L	M, H, K, F, R, Y, W
Polarity17	4.9–6.2	8.0–9.2	10.4–13.0
	L, I, F, W, C, M, V, Y	P, A, T, G, S	H, Q, R, K, N, E, D
Polarizability18	0–0.108	0.128–0.186	0.219–0.409
	G, A, S, D, T	C, P, N, V, E, Q, I, L	K, M, H, F, R, Y, W

Figure 2.8: Amino Acid Attributes of the Three Groups
The table defines the three groups, based on physico-chemical properties, behind the 21-dimensional vector. [17]

by a hydrophobic residue, or a hydrophobic residue by a polar residue; and (3) a neutral residue is followed by a hydrophobic residue, or a hydrophobic residue by a neutral residue. The D-features consist of the five numbers for each of the three groups, where each number is the fraction of the entire sequence, where the first residue of a given group is located, and where 25%, 50%, 75%, and 100% of those are contained. The complete parameter vector contains 21 scalar components.

Chou et al [15] use sequence-order features for protein subcellular localization, which highly concerns the protein function classification. Their method uses the physicochemical distance between amino acids to calculate the set of sequence-order-coupling numbers, which is used to reflect quasi-sequence-order effect. They use the physicochemical information of hydrophobicity, polarity and side chain volume. Shi et al [46] develop a different pseudo amino acid composition with the multi-scale energy approach. They use the same amino acid distribution and physicochemical information but different process method to generate the 50 feature values.

See Section 2.6.1 for how physico-chemical properties are used to determine SDS.

2.4 Profile Hidden Markov Model

The Markov model was proposed by Russian organic chemists Vladimir V. Markovnikov. The hidden Markov model (HMM), as a statistical analysis model, was developed in the 1920s. The hidden Markov model is a kind of Markov chain. Its state cannot be directly observed, but it can be observed by the observation vector sequence. Each observation vector is expressed in various states through probability distributions. Since the 1980s, it has been applied to speech recognition with great success. In recent years, it has also been widely studied and applied in the fields of bioinformatics.

For a random event, there is a sequence of observations, denoted as $O_1, O_2, ..., O_T$, in which a sequence of states is hidden: denoted $X_1, X_2, ..., X_T$. Three hypotheses are made in the hidden Markov model:

Markov hypothesis: The state constitutes a first-order Markov chain;

Immobility hypothesis: The state is independent of the specific event, and $p(X_{j+1}|X_j) = p(X_{i+1}|X_i)$ is established for arbitrary i, j;

Output independence hypothesis: The output is only relevant to the current state $P(O_1, O_2, ...O_t|X_1, X_2, ...X_t) = \prod_{i=1}^{t} p(O_i|X_i)$.

The Hidden Markov Model contains two stochastic processes, which are:

(1) the transitions between states described by the transition probability matrix, namely the **Markov chain**.

(2) the **general stochastic process**, using the observed value probability to describe the relationship between the state space and the observed value space.

The relationship between the observation value and the state is not a one-to-one correspondence, but a probability relationship.

The hidden Markov model is a five-tuple: $(\Omega_x, \Omega_o, A, B, \pi)$, where there is: a finite set of states $\Omega_x = \{q_1, q_2, ...q_N\}$; a finite set of observed values $\Omega_o = \{V_1, V_2, ...V_M\}$; a probability of transition $A = \{a_{ij}\}$; an emission probability $B = \{b_{ik}\}$, $b_{ik} = p(O_t = V_k|X_t = q_i)$; an initial

state distribution $\pi = \{\pi_i\}, \pi_i = P(X_1 = q_i)$. The meaning of the parameter in the hidden Markov model is shown in Table 2.5

Parameter	Meaning
N	Number of States
M	Number of Possible Observations Per State
A	Time-independent State Transition Probability Matrix
B	Probability Distribution of Observed Values in a Given State
π	Probability Distribution of Initial State Space

Table 2.5: The Meaning of the Parameter in the Hidden Markov Model.

The **profile Hidden Markov Model** contains left-right structures of the three states of matching state (M), insertion state (I) and missing state (D), which is shown in Figure 2.9. For the profile hidden Markov model, each node is a matching state (represented by a rectangle), an insertion state (represented by a diamond), a deleted state (represented by a circle), and a start state and an end (end) state, which do not emit any symbols. The match state indicates that the sequence has one character in the column; the delete state indicates that no characters are emitted in the column; the insert state allows additional characters to be emitted between the columns; therefore, each sequence passes through these states through the model from start to finish. Each column has a distribution of residues and a transition between states.

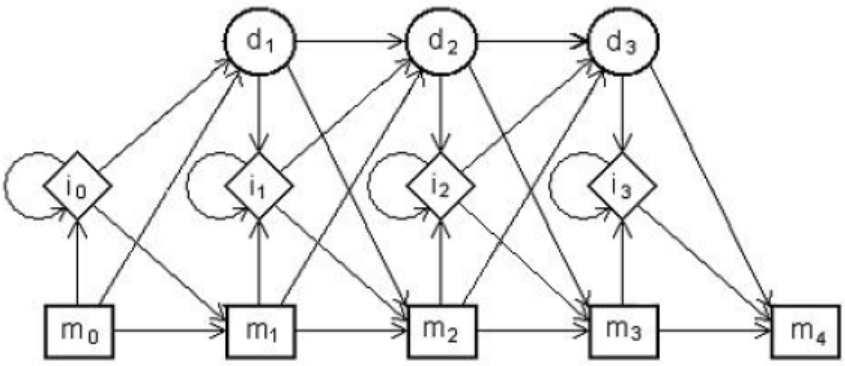

Figure 2.9: Profile Hidden Markov Model

A profile HMM showing matching states (M), insertion states (I) and missing states (D). A matching state is represented by a rectangle, an insertion state by a diamond, a deleted state a circle.

The main steps to create a profile HMM:

(1) Determine the matching status (main status). The gaps tend to be line up within one protein family multiple sequence alignment, and the model considers the ungapped regions. The probability of a new sequence x according to this model is $P(x|M) = \prod_{i=1}^{l} e_i(x_i)$. It is easier to evaluate the log-odds ratio $S = \sum_{i=1}^{l} log\frac{e_i(x_i)}{q_{x_i}}$, where q_{x_i} is the probability of the x under a random model. So there is an ungapped score matrices for the main status. The matrix is a position specific score matix (PSSM).

(2) Calculate the number of times the match status and the insert status symbol are issued and the number of transitions of various states.

(3) Convert the number of times the symbol is sent and the number of state transitions to the corresponding probability.

The most common problems solved by HMMs are:

(1) Full probability calculation problem, that is, given a sequence and a hidden Markov model, calculate the model to generate this sequence. This is a forward algorithm or a backward algorithm.

(2) The decoding problem, given the sequence and model, calculates the state chain that is most likely to generate this sequence. It uses the Viterbi algorithm.

(3) Learning problem, given a series of character sequences, and estimate the parameters of the hidden Markov model, which is using Welch-Baum algorithm.

2.4.1 Viterbi Algorithm

The **Viterbi algorithm** solves the problem of, given a known string, find the sequence most likely to produce the string; this is the **decoding problem**. An intuitive idea is to calculate the probability that all possible state sequences produce a string and then find the largest one. The amount of calculation of exponential growth makes this idea almost impossible to achieve. This problem can be solved with the idea of dynamic programming.

Defining Viterbi variables $\delta_t(k) = \max_{S_1,...,S_{K-1} \in S} P(C_1C_2...C_t, S_1S_2...S_{t-1}S_t = S_k)$, its intuitive meaning is along $S_1S_2...S_{t-1}S_t = S_k$, and the state at time t is a state chain of S_k, which

produces the maximum probability of a given string.

The Viterbi variable satisfies the following relationship: $\delta_t(k) = B(k, c_t) \times \max_{l=1,2,...,N}[\delta_l(t-1)A(l,k)], t = 2, ...n; k = 1, 2, ..., N$. In addition to calculating the probability, you also need to know which path is along. To do this, use a new variable to record the subscript for each max value.

Definition $\phi_{t+1}(S_k) = \underset{l=1,2,...,N}{arg\ max}[\delta_t(l)A(l,k)], t = 1, 2, ..., n-1, k = 1, 2, ..., N$. This variable is used to backtrack the maximum value subscript at each moment. Starting from the last moment n, step back and back, you can find the best path. If $s_n^* = \underset{l=1,2,...,N}{arg\max}[\delta_t(l)A(l,k)], s_t^* = \phi_{t+1}(S_{k+1}^*), t = n-1, ..., 1$, then the optimal path is $S^* = s_1^* s_2^* ... s_n^*$. The Viterbi algorithm returns the most probable alignment between a given sequence and the profile HMM. It finds the maximum probability path for the profile HMM to generate the query sequence. Let $V_{i,j}$ be the maximum probability of a path from start state S_i to the the end state S_j, so the $V_{i+1,j} = \max_{0 \le k \le j-1}(V_{i,k}P(k,j)P(q_{i+1}|j))$

$$V_j^M(i) = log\frac{e_{M_j}(x_i)}{q_{x_i}} + max\begin{cases} V_{j-1}^M(i-1) + loga_{M_{j-1}M_j} \\ V_{j-1}^I(i-1) + loga_{I_{j-1}M_j} \\ V_{j-1}^D(i-1) + loga_{D_{j-1}M_j} \end{cases}$$

$$V_j^I(i) = log\frac{e_{I_j}(x_i)}{q_{x_i}} + max\begin{cases} V_j^M(i-1) + loga_{M_jI_j} \\ V_j^I(i-1) + loga_{I_jI_j} \\ V_j^D(i-1) + loga_{D_jI_j} \end{cases}$$

$$V_j^D(i) = max\begin{cases} V_{j-1}^M(i) + loga_{M_{j-1}D_j} \\ V_{j-1}^I(i) + loga_{I_{j-1}D_j} \\ V_{j-1}^D(i) + loga_{D_{j-1}D_j} \end{cases}$$

where a_{ij} is the transition probability from state i to j and e_i is the emission probability in state i, and $V_j^M(i)$ is the log-odds score of the best path matching subsequence $x_{1,...,i}$ to the submodel up to state j, ending with x_i being emitted by state M_j. Similarly $V_j^I(i)$ is the score of the best path ending in x_i being emitted by I_j, and $V_j^D(i)$ is for the best path ending in state D_j.

2.5 Alignment Tools

2.5.1 BLAST

Protein functional sites are often composed of short sequence fragments, although mutations such as insertions and deletions may happen in other parts in the sequence, but these key functional sites are often quite conservative. Local alignments tend to be more sensitive to these functional segments than global alignments, so the results are more biologically significant. BLAST [2] (Basic Local Alignment Search Tool) is a program based on matching short sequence segments and has a powerful statistical model to determine the best local alignment of a query sequence against database sequences.

BLAST first filters the query sequence of low compositional complexity regions to prevent the production of large numbers of statistically significant but biologically uninteresting results. Then BLAST splits the query sequence into consecutive subsequences of lenght w, which is so called w-letter word pairs. BLAST is only interested in the high score word pairs, so there is a threshold T for the score of words. After compiling a list of word pairs at or above threshold T, the BLAST algorithm scans the word pairs against the database for sequences that match the words, which are called seeds. The seeds are the starting point of the ungapped local alignment between the query sequence and the database sequence. BLAST extends seeds to the right and left with a gapped extension [3]. The extension process is terminated when a score falls below a cutoff. BLAST reports the extended alignments or hits that have a score S at or above threshold and e-value E below threshold. Such! hits are called High Scoring Pairs (HSPs).

The substitution matrix is used to score the HSPs. So the higher the similarity, the higher the score S. The e-value E [37] is the probability (expected value) that BLAST program can randomly find a sequence with such a high score in the search space, so the higher the e-value, the more likely the result is randomly obtained and the less credible.

The program used the following parameters: Maximum target sequences: 10; E-value: 10; Word Size: 3; Substitution Matrix: BLOSUM 62; Gap Open Penalty: 11 as default; Gap Extension Penalty: 1 as default.

BLAST can be downloaded from: https://blast.ncbi.nlm.nih.gov/Blast.cgi?

CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=Download.

2.5.2 ClustalW

ClustalW [49] is a progressive alignment algorithm. Progressive alignment iteratively uses a pairwise dynamic programming alignment algorithm. It starts with the alignment of two sequences and gradually adds new sequences until all sequences are added. Different order of addition will produce different results. Thus determining the proper alignment order is a key issue for the progressive alignment algorithm. The more similar the two sequences are, the more confident people are in aligning them. So the alignment begins with the two most similar sequences.

In ClustalW, multiple sequences are paired to be aligned by dynamic programming approach [34]. For N sequences, there is $(N\text{-}1)(N)/2$ pairwise alignments with scores calculated using penalties for opening or extending gaps, and a substitution matrix. These scores are divided by the number of residues compared respectively, and generate a percent identity score. Both scores are converted to distances and generate a distance matrix. Second, using the distance matrix and neighbor-joining method [41], ClustalW builds a phylogenetic guide tree with topology (branching order and length). The progressive alignments follow the branching order in the guide tree. These trees are also used to derive a weight for each sequence. The weights are dependent upon the distance from the root of the tree, but sequences which have a common branch with other sequences share the weight derived from the shared branch [49].

The default parameters are: Gap Open Penalty: 10.0; Gap Extension Penalty: 0.1; Substitution Matrix: BLOSUM 62.

ClustalW can be downloaded from: http://www.clustal.org/clustal2/.

2.5.3 Clustal Omega

Clustal Omega [47] is another progressive alignment method from the Clustal family. The key to making the progressive alignment is the method used to make the guide tree. Clustal Omega uses a modified version of mBed [7]. Each sequence is embedded in a vector space of n dimensions, where each element is the distance to one of n reference sequences. The pair-wise alignments are computed using the very accurate HHalign package [45], which aligns sequences by two profile

hidden Markov models [18].

Clustal Omega can be downloaded from: http://www.clustal.org/omega/.

2.5.4 MAFFT

MAFFT [27] is a multiple sequence alignment method based on fast Fourier transform. It treats sequence information as a signal for processing. The frequency of amino acid substitutions strongly depends on the difference of physico-chemical properties. Each amino acid is converted into a vector, which includes two values: volume and polarity. The algorithm calculates the relationship between two sequences, the result is transformed to Fourier signal information. This transform can reduce the cpu time. If two sequences compared have homologous regions, there will be a peak corresponding to these regions. MAFFT applies standard dynamic programming to obtain an optimal path, which corresponds to the optimal arrangement of similar segments. MAFFT is an iterative method. In the program package MAFFT, there are several different modes: progressive methods FFT-NS-1 and FFT-NS-2, and an iterative refinement method FFT-NS-i. FFT-NS-1 uses a guide tree and progressively aligns ! the sequences. FFT-NS-2 repeats the alignments on the base of FFT-NS-1. FFT-NS-i divides the alignment into two groups and realigns until there is no improvement in score.

MAFFT can be downloaded from: http://mafft.cbrc.jp/alignment/software/.

2.5.5 T-Coffee

T-Coffee [35] is the short name which stands for “tree-based consistency objective function for alignment evaluation”. T-Coffee is also a progressive alignment algorithm, which is basically similar to the algorithm step of ClustalW. The difference between them is that, in T-Coffee, extension library is used instead of the substitution matrix in ClustalW. So the scoring information used in each step of the progressive alignment process is taken from the relationship within all sequences, not just the sequence that is currently being aligned. hence, it is consistency-based. This minimizes the greedy effects of the progressive alignment algorithm. Since both the local alignment and the global alignment are considered in the extension library, all the pair-wise sequences alignment data is preprocessed, and the result is more accurate.

T-Coffee algorithm has five steps. The first one is generating a primary library. In this step, T-Coffee uses two alignment sources for each pair of sequences, one is local (Lalign [26]) and the other one is global (ClustalW [49]). The second step is the derivation of the primary library weights. There is a weight which is generated on each pair of aligned residues, and it is equal to sequence identity within the pair-wise alignment. So there are two libraries (local and grobal) for each set of sequences. Third, T-Coffee combines these two libraries by merging the duplicated pair-wise residues and adding the sum of the two weights. The fourth step is extending the library. This enormously increases the value of the information in the library by examining the consistency of each pair of residues with residue pairs from all of the other alignments. A triplet approach is used. For example, for sequences A, B and C, where A(G) aligned with B(G), and there is a! primary weight, then the weight(A(G), B(G)) will be changed depending on the weight(A(G), C(G)) and weight(B(G), C(G)). The specific changes are described in detail in the paper [35]. Lastly, the progressive alignment strategy uses the the extension library as substitution matrix to align the sequences according to the phylogenetic tree [41].

T-Coffee can be downloaded from: http://www.tcoffee.org/Projects/tcoffee/index.html#DOWNLOAD.

2.5.6 MUSCLE

MUSCLE [20] is analgorithm which combines the progressive method and the iterative method. The steps are shown in Figure 2.5.6. There are two parts in the algorithm. The first part uses two approaches to calculate the distances between sequences. The first distance calculation is by kmer distance (for an aligned pair). By progressive alignment, MUSCLE makes a preliminary alignment, MSA1. The second distance is Kimura distance (for an aligned pair), which is generated from the pairwise identity of MSA1. Then it builds another progressive alignment MSA2. The guide trees of these two progressive alignments are both built by the UPGMA method. In the second part, MUSCLE uses a group aligning idea to do iterative alignment. According to the branching point of the second guide tree, the sequences are divided into two groups. The profile of the multiple sequence alignment in each subtree is computed. Then MUSCLE uses log-expectation to calculate ! the two profile scores. A new alignment MAS3 is made, and if the score is improved, the new

alignment is kept, otherwise it is discarded.

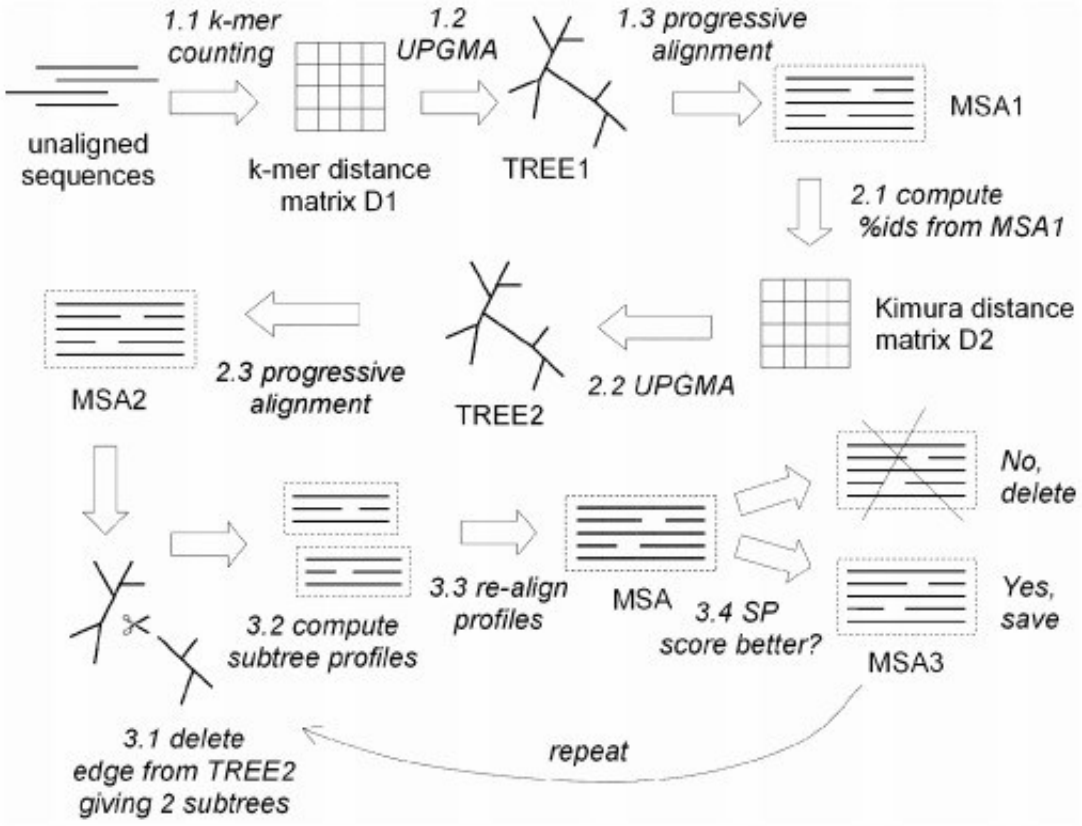

Figure 2.10: MUSCLE Algorithm

The steps of the MUSCLE algorithm showing the three multiple sequence alignments MSA1, MSA2, MSA3, and their relationship. [20]

MUSCLE can be downloaded from: https://www.drive5.com/muscle/downloads.htm.

2.5.7 TM-Coffee

Membrane-bound proteins are a special class of proteins. The regions that insert into the cell-membrane have a profoundly different hydrophobicity pattern compared with soluble proteins. Multiple alignment techniques use scoring schemes tailored for soluble proteins and are therefore, in principle, not optimal to align membrane-bound proteins [39]. TM-Coffee [22] is designed for this situation by using homology extension. Although this method gives much more accurate alignment,

performing an alignment takes several orders of magnitude longer than the standalone applications [21].

The algorithm of TM-Coffee follows. First, run BLAST against a specified database for each sequence that is going to be aligned. Then TM-Coffee turns the BLAST output into a profile by removing all columns corresponding to positions unaligned to the query (i.e. gaps in the query) and the query positions unmatched by BLAST are filled with gaps [22]. Third, TM-Coffee produces a T-Coffee library by aligning every pair of profiles with a pair-HMM, and every pair of matched columns with a posterior probability of being aligned higher than 0.99 is added to the library. Last, this library is uesd to do the progressive alignment.

TM-Coffee can be downloaded from: http://www.tcoffee.org/Projects/tcoffee/index.html#DOWNLOAD.

2.6 Tools for Specificity Determining Sites

2.6.1 SPEER SERVER

SPEER is an ensemble approach that encodes the conservation patterns of amino acid types using their physicochemical properties and the heterogeneity of evolutionary changes between and within the subfamilies [13].

The first SPEER term, which accounts for the variability of sites between and within the subfamilies, is the Euclidean distance based on amino acid physico-chemical properties. SPEER uses matrices containing quantitative values for amino acid physico-chemical properties which were obtained from the UMBC AAIndex database [8]. To quantify the difference between any two sequences p and q at a given site, SPEER uses a weighted Euclidean distance (ED). The ED score is positive and low values correspond to the situation where amino acid properties are very well conserved within the subfamilies and non-conserved between them. The second SPEER term is evolutionary rate. SPEER uses the approach implemented in the rate4Site [40] program to calculate the evolution rate (ER) at each site separately for each subfamily and then average it among all subfamilies. A low average ER value would indicate that there is a slowly evolving site in certain

subfamilies. The third SPEER term is combined relative entropy. SPEER calculates relative entropy for each pair within one subfamily, and gets the combined relative entropy (CRE). A large CRE value corresponds to large differences between amino acid distributions within the subfamily.

This experiment used the weights of 1.0, 1.0 and 1.0 for the parameters `relative entropy`, `PC property distance` and `ER`, respectively.

Speer Server can be download from: http://www.hpppi.iicb.res.in/ss/download.html.

2.6.2 SimGroup

GroupSim [11] uses a score-based approch. In GroupSim, the average similarity between each pair of amino acids within a subgroup is calculated using a similarity matrix to obtain a column score, based on which the SDS predictions are done [14]. A protein family may be grouped into subfamilies that share specific functions that are not common to the entire family. Often, the amino acids present in a small number of sequence positions would determine each protein's particular functional specificity. GroupSim presents a sequence-based method to predict this kind of SDS.

First, it calculates the average similarity between each pair of amino acids in a subfamily according to a similarity matrix for each subfamily. Second, GroupSim computes, for each subfamily, the average similarity of all amino acid pairs containing one amino acid in the group and one not in the group. When the subfamilies are more different, the column is more likely to be a SDS. Last, the column score is the average within-subfamily similarity minus the average between-subfamily similarity. Higher scores mean a greater likelihood to be a SDS.

GroupSim can be download from: http://compbio.cs.princeton.edu/specificity/.

2.6.3 Xdet

Xdet [36] implements two methods: one is the mutational behaviour (MB) method, and the second one is a version of the MB-method, which uses an external arbitrary functional classification instead of relying on the one implicit in the alignment. It detects residues responsible for functional specificity in multiple sequence alignments. This kind of site, representing a family-dependent (or function-dependent) conservation pattern, complements the fully-conserved positions as predictors

of functionality.

For each position in the alignment, there is a matrix generated to calculate the amino acid changes for all pairs of proteins based on the substitution matrix. Each entry in this matrix represents the similarity between the residues of two proteins at that position. Second, another matrix is constructed by an external arbitrary functional classification. Each entry in this matrix represents the functional similarity between the corresponding proteins. Last, Xdet uses the Spearman rank-order correlation coeficient to compare these two matrices:

$$r_k = \frac{\sum_{pq}(A_{pqk}{}' - \overline{A}')(F_{pq}{}' - \overline{F}')}{\sqrt{\sum_{pq}\left(A_{pqk}{}' - \overline{A}'\right)^2}\sqrt{\sum_{pq}\left(F_{pq}{}' - \overline{F}'\right)^2}} \tag{6}$$

where r_k is the score for position k, A_{pqk} is the similarity between the amino acids of proteins p and q at position k, and F_{pq} is the functional similarity between proteins p and q. Positions with high r_k values are the ones for which similarities between amino acids are correlated with the functional similarities between the corresponding proteins, and hence are predicted as the ones related with functional specificity [36].

Xdet can be download from: `http://pdg.cnb.uam.es/pazos/Xdet/`.

2.7 HMMER for Profile Hidden Markov Models

HMMER [19] is a free and commonly used software package for sequence analysis based on profile Hidden Markov Models. For the multiple sequence alignment of the protein sequences in a protein family, there are functional sites, which are corresponding to conserved amino acid sites; and non-specific feature sites, which are corresponding to less conserved amino acid sites. So each site has a different probability distribution over 20 amino acids, which measures the likelihood of each amino acid occurring at that site in the protein family. Multiple sequence alignments can then be modeled by capturing a probabilistic model of the shared nature of multiple sequence alignments [30].

In HMMER, the application `hmmbuild` is used to build a profile HMM using a multiple sequence alignment, or single sequence as input. The hidden state is one of insert state, delete state

and match state, while the observation layer consists of sequence letters, as shown in Figure 2.9. The transition probability and emission probability are calculated from the given multiple sequence alignment of the protein sequences, and a profile HMM for a protein family is built. The second application is `hmmscan`, which scans a single protein sequence against a database of profile HMMs.

The program parameters used are: e-value: 0.1.

HMMER can be run download from: `http://hmmer.org/download.html`.

2.8 The State of the Art

2.8.1 Helms Lad

Schaadt et al. [44] used amino acid composition, characteristics of amino acid residues and conservation to detect transporters based on different substrates, amino acids, oligopeptides, phosphates and hexoses and showed an accuracy of 75% to 90%. They classified to four substrate categories: amino acid, oligopeptide, phosphate, and hexose. The number of characterized transporters in A. thaliana for the four substrates numbered from 13 to 17. They constructed a vector for each protein using various types of amino acid composition, AAC, PAAC, PseAAC, PsePAAC, MSA-AAC, and used Euclidean distance from the query protein's vector to the known vectors to rank the substrate categories. They found that AAC did not yield accurate results. However, PAAC performed as well as the more complicated PsePAAC and MSA-AAC, yielding accuracy over 90%.

2.8.2 Zhao Lab

TrSSP (Transporter Substrate Specificity Prediction Server) [33] is a web server to predict membrane transport proteins and their substrate category. The substrate categories are: (1) oligopeptides (amino acid); (2) anion; (3) cation; (4) electron; (5) protein/mRNA; (6) sugar; and (7) other. TrSSP makes a top-level prediction of whether the protein is a transporter, or not. A SVM is applied with highest accuracy being reported using Amino Acid index (AAindex) and Position-Specific Scoring Matrix (PSSM).

Chapter 3

Constructing and Evaluating Classifiers

Several existing classifiers target the classification of substrates, however, they are able to classify the type of the substrates being transported, but not the specific substrate. There are not enough examples of transporters where the specific substrates are known — that is, have been experimentally determined — to allow machine learning approaches to achieve good perfromance. Therefore, we propose a new approach to classify the class of substrates transported across a membrane by a given transmembrane protein. We develop a classification pipeline which integrates the steps of data rocessing, model building and model evaluation. The pipeline contains similarity search, multiple sequence alignment (MSA), specificity determining site (SDS) prediction and construction of a profile Hidden Markov Model. We evaluate the pipeline across a combination of existing MSA tools and SDS tools. We discuss and analyze the impact of the using different MSA tools, SDS tools and parameters on the prediction of the substrate class. We find a combination which outperforms the state-of-the-art.

This work is the first to apply profile Hidden Markov Models to classify the substrate class of a transmembrane transport protein, and the first to combine the specificity determining sites with profile Hidden Markov Model to the classification for the substrate class transported across a membrane by a given transmembrane transport protein.

This chapter is organized as follows: Section 3.1 presents the material used in the project, which includes the dataset and tools whaih are called during the experiment; Section 3.2 specifically describes the experimental procedure. Section 3.3 shows the experimental results; Section 3.4

discuesses the experimental results.

3.1 Materials

3.1.1 Datasets

We used two datasets in this work

(1) The small dataset [44] has 61 sequences and covers four substrate classes. See Table 3.1.

(2) The large dataset [33] has 780 sequences in the training set and 120 sequences in the test set, and which covers seven substrate classes, one of which is *"Other"*. See Table 3.2.

Complete lists of the sequences for each dataset are available in he appendices.

Substrate Class	Set Size	Number of TCDB Families	TCDB Families
Amino Acid	14	2	2.A.18, 2.A.3
Hexose	15	2	2.A.1.1, 2.A.123
Oligopeptide	17	2	2.A.67, 2.A.17
Phosphate	15	5	2.A 1.9, 2.A.1.14 2,A.20, 2.A.29, 2.A.7

Table 3.1: Dataset 1.

Substrate Class	Training Dataset Size	Test Dataset Size
Amino Acid	70	15
Anion	60	12
Cation	260	36
Electron	60	10
Protein/mRNA	70	15
Sugar	60	12
Other	200	20
Total	780	120

Table 3.2: Dataset 2.

3.1.2 Databases

We use the Swiss-Prot database when searching for similar sequences in BLAST and TM-Coffee. Since both datasets consist of sequences from Swiss-Prot, we remove them from the database used for similarity searches.

3.2 Methods

We use a classification pipeline which integrates the steps of data rocessing, model building and model evaluation. The pipeline contains similarity search, multiple sequence alignment (MSA), specificity determining site (SDS) prediction and construction of a profile Hidden Markov Model.

We evaluate the pipeline across a combination of existing MSA tools and SDS tools. We discuss and analyze the impact of the using different MSA tools, SDS tools and parameters on the classification of the substrate class. We find a combination which outperforms the state-of-the-art.

3.2.1 BLAST

The first step of our experiment is to run BLAST on each sequence in training dataset to get similar sequences. BLAST need a local database, which is made of Swiss-Prot by removing any copy of sequences in the test dataset. For each sequence s, a BLAST search is performed to retrieve a maximum of 10 similar sequences.

Algorithm 1 BLAST

Input: the training dataset as .fasta file of protein sequences;
Input: the test dataset as .fasta file of protein sequences;
Output: result is up to 10 similar sequences for each sequence in the training datase;t

1: LocalDB $\Leftarrow$ Search(Swiss-Prot, removing any copy of sequences in the test dataset)
2: **for** each sequence s in training dataset **do**
3: SimilarSeq[s] $\Leftarrow$ blastp(DB=LocalDB, query=s, maxseq=10)

Our BLAST command was the following:

```
BLAST blastp -query blast_query_path -db cleaned_swiss_prot_path \
-evalue 0.001 -outfmt 5 -out des_blast_file -max_target_seqs 10
```

3.2.2 MSA

We compute a MSA of the set SimilarSeq[s] of similar sequences to s, using an MSA tool, such as ClustalW. Algorithm 2 shows the process of multiple sequence alignment.

We also use a variety of different MSA tools: ClustalW, Clustal Omega, MAFFT, T-Coffee, MUSCLE and TM-Coffee. Figure 3.1 shows the process of using different MSA tools to align the

sequences.

Algorithm 2 MSA

Input: the sequences $S = s$ and SimilarSeq[s];
Output: result is an alignment MA of the sequences;
1: MA $\Leftarrow$ ClustalW(S)

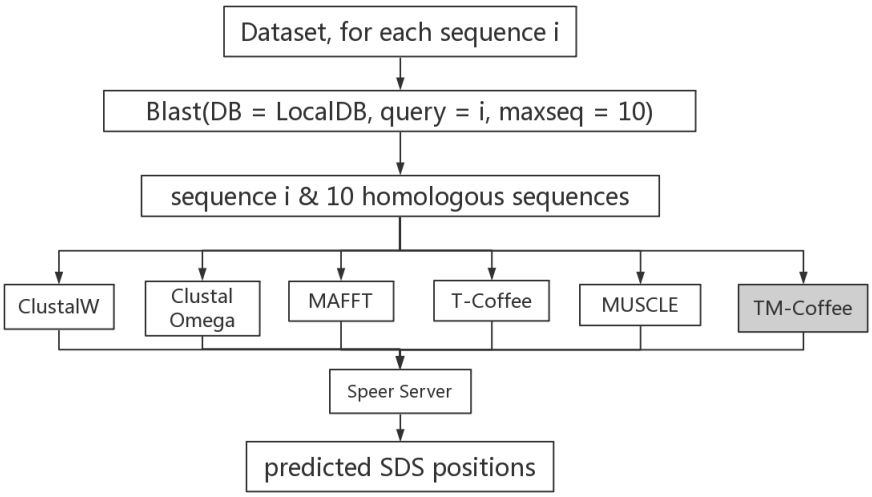

Figure 3.1: The method showing use of various MSA tools.

Our MSA command was the following:

```
ClustalW

clustalw2 -infile=clustalw_input_path -outfile=clustalw_output_path \

-OUTPUT=FASTA


Clustal Omega

clustalo -infile=input_path -outfile=output_path


MAFFT

mafft input_path  > output_path


T-Coffee
```

```
t_coffee tcoffee_input_path -outfile tcoffee_output_path -output fasta
```

```
MUSCLE
muscle -in input_path -out output_path
```

```
TM-Coffee
t_coffee tm_coffee_input_path -outfile tm_coffee_output_path \
-output fasta -mode psicoffee -blast_server LOCAL -protein_db \
cleaned_swiss_prot_path
```

3.2.3 SDS

The prediction of SDS is used to filter an MSA, and focus on the important posiions in the sequence when building a profile HMM. We process the alignment, so all entries in the columns of non-SDS positions are changed to '-', and other entries are unchanged. Algorithm 3 shows the process to predict the SDS from the MSA, and to filter the MSA.

We also use a variety of different SDS tools: Speer Server, GroupSim, XDet, as shown in Figure 3.2.

Algorithm 3 SDS

Input: an MSA MA;
Output: result is fMA, a filtered MSA;

1: SDS $\Leftarrow$ SpeerServer(MSASeq)
2: $fMA \Leftarrow MA$
3: **for** each position p not in SDS **do**
4: $\quad fMA[r, p] \Leftarrow$ '-', for each row r of alignment

Our SDS command was the following:

```
Speer Server
./SPEER -wRE 1 -wEDist 1 -wERate 1 \
-i input_path -o output_path 6 5
```

```
GroupSim
```

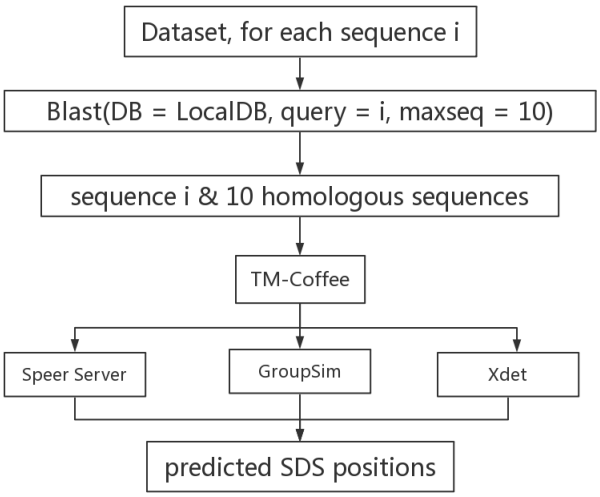

Figure 3.2: The method showing the use of various SDS tools.

```
python groupsim.py -n -m metric_path -w 3 -o output_path \
input_path group1 group2
```

```
Xdet
./xdet_osx input_path metrix_path > output_path -S 10
```

3.2.4 Build HMM

A profile HMM is built by HMMER's `hmmbuild` from the filtered MSA as shown in Algorithm 4.

Algorithm 4 HMM

Input: fMA, a filtered or non-filtered MSA;
Output: result is a profile HMM;
1: profile HMM $\Leftarrow$ hmmbuild(fMA)

Our HMMBuild command was the following:

```
HMMBuild
./hmmbuild output_path input_path
```

3.2.5 Classification

Classification of a test sequence p is done by running the sequence against each profile HMM, selecting the one with the bext score, and selecting the corresponding substrate class. (See Algorithm 5). The score is computed using HMMER's `hmmscan`. This requires that the profile HMM be compressed, which is done by `hmmpress`.

Algorithm 5 Classification

Input: a sequence p of a transport protein to classify;
Input: a collection of profile HMMs;
Output: classification of p;

1: **for** each profile HMM **do**
2: compressed profile HMM $\Leftarrow$ hmmpress(profile HMM)
3: **for** each compressed profile HMM m **do**
4: score[m] $\Leftarrow$ hmmscan(m,p)
5: classify p to the substrate class corresponding to m which generated the highest score

Our HMMScan command was the following:

```
HMMScan
./hmmscan -o result_path -E 0.1 hmm_path query_path
```

3.2.6 Evaluation

Four statistical measures were considered to measure the performance:

sensitivity which is the proportion of positives that are correctly identified:

$$Sensitivity = \frac{TP}{TP + FN} \tag{7}$$

specificity which is the proportion of negatives that are correctly identified:

$$Specificity = \frac{TN}{TN + FP} \tag{8}$$

accuracy which is proportion of correct classifications made amongst all the classifications:

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{9}$$

Matthews correlation coefficient (MCC) which is a single measure taking into account true and false positives and negatives:

$$MCC = \frac{(TP \times TN - FP \times FN)}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \quad (10)$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

We use the MCC because it is less influenced by imbalanced data and is arguably the best single assessment metric in this case [16, 50, 6].

3.2.7 Experiments

We performs experiments to analyze the impact of the using different MSA tools, SDS tools and parameters on the classification of the substrate class.

For each dataset, we consider each combination of MSA tool and SDS tool to determine the bst combination. We determine the impact of the use of MSA, and the use of SDS by including baseline pipelines. The primary baseline uses no MSA tool and no SDS tool, and simply builds a profile HMM from the sequence itself. For each MSA tool, we have a baseline pipeline that uses no SDS tool.

We perform experiments to guage the effect of the parameter settings for the SDS tools.

For our best perfroming pipeline, we do a comparison against the state-of-the-art TrSSP [33] using the test set from the second dataset.

3.3 Results

3.3.1 First Dataset

Table 3.3 presents the overall accuracy and MCC of the 25 combinations, including the baselines. They are sorted by Accuracy, from low to high, as the MCC is consistently 1.00. The highest accuracy was achieved by TM-Coffee-GroupSim, so we present its detailed performance in Table 3.4, nd its confusion matrix in Table 3.5.

MSA	SDS	Accuracy	MCC
T-Coffee (baseline)	-	0.951	1.00
ClustalW (baseline)	-	0.967	1.00
ClustalOmega	NO-SDS	0.967	1.00
MUSCLE (baseline)	-	0.967	1.00
ClustalW	SpeerServer	0.980	1.00
MAFFT	SpeerServer	0.980	1.00
TM-Coffee	SpeerServer	0.980	1.00
MAFFT (baseline)	-	0.984	1.00
TM-Coffee (baseline)	-	0.984	1.00
ClustalOmega	SpeerServer	0.984	1.00
T-Coffee	SpeerServer	0.984	1.00
MUSCLE	SpeerServer	0.984	1.00
T-Coffee	GroupSim	0.984	1.00
T-Coffee	Xdet	0.984	1.00
- (baseline)	-	0.988	1.00
ClustalOmega	Xdet	0.988	1.00
MUSCLE	Xdet	0.988	1.00
TM-Coffee	Xdet	0.988	1.00
ClustalW	Xdet	0.992	1.00
ClustalOmega	Xdet	0.992	1.00
ClustalW	GroupSim	0.996	1.00
ClustalOmega	GroupSim	0.996	1.00
MAFFT	GroupSim	0.996	1.00
MUSCLE	GroupSim	0.996	1.00
TM-Coffee	GroupSim	0.996	1.00

Table 3.3: Overall performance of the combinations (First dataset).
The results come from the Leave-One-Out-Cross-Validation test using the first dataset. The combinations are ranked by Accuracy. Note the baseline entries.

Class	Sensivity	Specificity	Accuracy	MCC	Unclassified
Amino Acid	1.000	1.000	1.000	1.000	0.000
Hexose	1.000	1.000	1.000	1.000	0.000
Oligopeptide	1.000	1.000	1.000	1.000	0.000
Phosphate	0.933	1.000	0.984	1.000	0.067
Overall			0.996	1.000	0.016

Table 3.4: Detailed performance of TM-Coffee-GroupSim combination (First dataset).
The results come from the Leave-One-Out-Cross-Validation test using the first dataset. The column "Unclassified" records the percentage of sequences that were a match to any profile HMM.

Actual \ classified	Amino Acid	Hexose	Oligopeptide	Phosphate	Unclassified
Amino Acid	14	0	0	0	0
Hexose	0	15	0	0	0
Oligopeptide	0	0	17	0	0
Phosphate	0	0	0	14	1

Table 3.5: The Confusion matrix of TM-Coffee-GroupSim combination (First dataset). The results come from the Leave-One-Out-Cross-Validation test using the first dataset. The column "Unclassified" records the number of sequences that were a match to any profile HMM.

3.3.2 Second Dataset

Table 3.6 presents the overall accuracy and MCC of the 25 methods, including the baselines. Table 3.3 presents the overall accuracy and MCC of the 25 combinations, including the baselines. They are sorted by MCC, from low to high. The highest MCC was achieved by MUSCLE and XDet, so we present its detailed performance in Table 3.7, and its confusion matrix in Table 3.8.

3.3.3 Parameters

To make the impact of SDS on the classification results, we set different parameters for the SDS tools.

Table 3.9 and Table 3.10 shows the impact of the SDS threshold for the first dataset. The first table in terms of actual counts, and the second table in terms of the performance metrics.

Table 3.11 shows the result of varying parametesr for Speer Server. There are three parameters in Speer Server: `relative entropy` (-wRE), `evolutionary distance` (-wDist), `evolutionary rate` (-wERate). The program uses a linear combination of three terms, each with a weight between 0 and 1, with at least one having a non-zero value.

For GroupSim, there are two parameters: the substitution matrix, and the conservation window size. Varying the window size did not affect the results. For XDet, there are two parameters: the substitution matrix, and random alignments. the random alignments are used to associate Z-scores and P-values to the correlation scores. We tried generating 1, 5 and 10 random alignments, however, it did not affect the result.

MSA	SDS	Accuracy	Average Accuracy	MCC
MAFFT	SpeerServer	0.508	0.914	0.591
MUSCLE	SpeerServer	0.517	0.914	0.593
ClustalOmega	SpeerServer	0.525	0.914	0.593
TM-Coffee	SpeerServer	0.533	0.918	0.605
ClustalW	SpeerServer	0.525	0.919	0.612
T-Coffee (baseline)	-	0.608	0.925	0.636
T-Coffee	GroupSim	0.608	0.926	0.638
T-Coffee	Xdet	0.608	0.926	0.638
T-Coffee	SpeerServer	0.600	0.925	0.639
MAFFT	GroupSim	0.650	0.931	0.668
TM-Coffee	Xdet	0.683	0.936	0.675
- (baseline)	-	0.650	0.933	0.676
ClustalOmega	Xdet	0.667	0.933	0.681
ClustalOmega (baseline)	-	0.683	0.933	0.683
ClustalW	GroupSim	0.667	0.935	0.685
MAFFT (baseline)	-	0.683	0.935	0.686
ClustalOmega	GroupSim	0.667	0.933	0.688
TM-Coffee	GroupSim	0.667	0.933	0.689
MUSCLE	GroupSim	0.675	0.936	0.690
ClustalW	Xdet	0.683	0.936	0.692
ClustalW (baseline)	-	0.692	0.936	0.693
TM-Coffee (baseline)	-	0.692	0.937	0.697
MAFFT	Xdet	0.683	0.938	0.702
MUSCLE (baseline)	-	0.700	0.938	0.703
MUSCLE	Xdet	0.700	0.940	0.716

Table 3.6: Overall performance of the combinations (Second dataset).
These are the results obtained using the independent test set of size 120. The combinations are ranked by MCC. For each combination, the table presents accuracy (calculated accuracy as proportion of correct predictions divided by the total number of predictions), average accuracy (calculated as the average across the seven classes) and MCC. Note the baseline entries.

Class	Sensivity	Specificity	Accuracy	MCC	Unclassified
Amino Acid	0.73	0.98	0.95	0.76	0.07
Anion	0.67	0.96	0.93	0.63	0.00
Cation	0.86	0.99	0.95	0.88	0.14
Electron	0.50	1.00	0.96	0.69	0.50
Protein	0.60	1.00	0.95	0.75	0.40
Sugar	0.75	0.97	0.95	0.72	0.17
Other	0.55	0.96	0.89	0.57	0.15
Overall			0.70	0.72	0.21

Table 3.7: Detailed performance for the MUSCLE and Xdet combination (Second dataset).
The results come from the independent test set of size 120. The column "Unclassified" records the number of sequences that were a match to any profile HMM.

Actual \ classified	Amino Acid	Anion	Cation	Electron	Protein	Sugar	Other	Unclassified
Amino Acid	11	0	1	0	0	1	1	1
Anion	1	8	0	0	0	1	2	0
Cation	0	0	31	0	0	0	0	5
Electron	0	0	0	5	0	0	0	5
Protein	0	0	0	0	9	0	0	6
Sugar	0	0	0	0	0	9	1	2
Other	1	4	0	0	0	1	11	3

Table 3.8: Confusion matrix for the MUSCLE and Xdet combination (Second dataset).
The results come from the independent test set of size 120. The column "Unclassified" records the number of sequences that were a match to any profile HMM.

Threshold	TP	TN	FP	FN	SDS/length
0	60	183	0	1	27475 / 34588
0.05	60	183	0	1	27109 / 34588
0.10	60	183	0	1	25953 / 34588
0.15	59	183	0	2	23064 / 34588
0.20	59	183	0	2	18142 / 34588
0.25	36	183	0	25	12276 / 34588
0.30	0	183	0	61	7799 / 34588

Table 3.9: Different thresholds for SDS tools (First dataset).
This is the result of the combination of TM-Coffee and GroupSim. The first column, threshold, is set for the score given by GroupSim. The GroupSim score is 0 or 1. The threshold 0 means that returning the sites that has a GroupSim score higher than 0. The last column, "SDS/length", is respectively the number of SDS and the length of the MSA.

Threshold	Sensitivity	Specificity	Accuracy	MCC	Unclassified
0	0.984	1.000	0.996	1.000	0.016
0.05	0.984	1.000	0.996	1.000	0.016
0.10	0.984	1.000	0.996	1.000	0.016
0.15	0.967	1.000	0.992	1.000	0.033
0.20	0.967	1.000	0.992	1.000	0.033
0.25	0.590	1.000	0.898	1.000	0.410
0.30	0.000	1.000	0.000	1.000	0.000

Table 3.10: Performance for different thresholds for SDS tools (First dataset).

Parameter			Sensivity	Specificity	Accuracy	MCC	Unclassified
RE	Dist	ERate	Sensivity	Specificity	Accuracy	MCC	Unclassified
1	1	1	0.685	0.000	0.600	0.639	0.314
0	1	1	0.685	0.000	0.600	0.639	0.314
1	0	1	0.707	0.000	0.625	0.661	0.292
1	1	0	0.685	0.000	0.600	0.639	0.314

Table 3.11: Performance for different parameters of Speer Server (Second dataset).

3.3.4 Comparison with State of the Art

Table 3.12 compares the performance of the MUSCLE-Xdet combination to the state-of-the-art TrSSP [33] on the independent test set of size 120 from the second dataset.

Class	Specificity		Sensitivity		Accuracy		MCC	
	TrSSP	MUSCLE -Xdet	TrSSP	MUSCL -Xdet	TrSSP	MUSCLE -Xdet	TrSSP	MUSCLE -Xdet
Amino acid	82.42	98.10	93.33	73.33	83.33	95.00	0.49	0.76
Anion	69.05	96.30	75.00	66.67	69.44	93.33	0.23	0.63
Cation	74.31	98.81	75.00	86.11	74.44	95.00	0.41	0.88
Electron	91.78	100.00	80.00	50.00	91.11	95.83	0.50	0.69
Protein	82.42	100.00	93.33	60.00	83.33	95.00	0.49	0.75
Sugar	76.79	97.22	91.67	75.00	77.78	95.00	0.38	0.72
Other	73.13	96.00	60.00	55.00	71.67	89.17	0.23	0.57
Overall					78.88	94.04	0.41	0.72

Table 3.12: Comparison between the MUSCLE-Xdet combination and TrSSP (Second dataset). The table presents our data for MUSCLE-Xdet built with the training set and run on the test set. The corresponding results for TrSSP are taken from their original paper. The table shows specificity, sensitivity, accuracy and MCC for each of the seven substrate types; and the average accuracy and MCC. We calculated accuracy as proportion of correct predictions divided by the total number of predictions, and MCC from the confusion matrix as in Equation 10. The TrSSP results were calculated as the average across the seven classes; if we adopt the same method the average accuracy is 94.04% and the average MCC is 0.72.

3.3.5 The number of sequences returning from BLAST

Table 3.13 shows the number of sequences returned by BLAST in the first and second dataset. Table 3.14 shows the sequences which returning zero BLAST hits in the second training dataset.

Number	Number of Sequences in First Dataset	Number of Sequences in Second Dataset
10	61	689
9	0	6
8	0	5
7	0	9
6	0	6
5	0	13
4	0	15
3	0	10
2	0	9
1	0	12
0	0	6

Table 3.13: The number of sequences returned by BLAST in the first and second dataset.

Substrate Class	Sequences ID
Anion	Q9LFX9
Cation	Q63089 D5AQY6 P48836
Electron	P0AF32
Other	P02929

Table 3.14: The sequences which returning zero BLAST hits in the second training dataset.

3.4 Discussion

3.4.1 Outperforming the State of the Art

Table 3.12 compares the performance of the MUSCLE-Xdet combination to the state-of-the-art TrSSP [33]. It shows that we outperform TrSSP on each individual substrate class in terms of both accuracy and MCC. Overall, our average MCC is 0.72 compared to 0.41 for TrSSP.

TrSSP does have better sensitivity than our best performing combination, which may be due to the number of unclassified sequences.

3.4.2 Combination of MSA Tools and SDS Tools

For the first dataset, Table 3.3 shows the performance for the combination of MSA and SDS tools. The best performing combinations based on accuracy are ClustalW-GroupSim, ClustalOmega-GroupSim, MAFFT-GroupSim, MUSCLE-GroupSim and TM-Coffee-GroupSim with an accuracy (99.6%) compare to the accuracy of the baseline of 98.8%. For the second dataset, Table 3.6 shows the performance of the combinations of MSA and SDS tools. Here they are ranked by MCC, which is the preferred metric for the imbalanced datasets that we have. The best performing combination is MUSCLE and Xdet with an MCC of 0.716 compared to the baseline of 0.676.

So we can conclude there is a benefit to including SDS tools in the pipeline with MSA, and there is a benefit to including MSA tools in the pipelne.

We believe that the improved performance is due to the MSA tools incorporating the evolutionary information, and the SDS tools identifying important positions in the protein.

We expected combinations with TM-Coffee to outperfrom the other MSA tools, because TM-Coffee produces alignments consistent with the transmembrane segments, but this was not the case.

3.4.3 Parameters

The results of our experiments with various parameter setting for the SDS tools showed that they had no impact on performance of the classifiers, in general. Setting the threshold, as shown in Table 3.10, does impact the accuracy. We can clearly observe that as the number of SDS decreases, the accuracy of the classification also decreases.

3.4.4 Unclassified Sequences

The number of unclassified sequences was small for both the first and second datasets when we looked at each combination of tools, so we checked whether it was the same sequences turung up each time. It was, though we have not done further analysis to see why that is the case.

Table 3.15 show the unclassified sequences from the first dataset for combinations of the different MSA tools and Speer Server. Table 3.16 shows the unclassified sequences from the second dataset for combinations of the different MSA tools and XDet.

Substrate	Sequence	ClustalW	Clustal Omega	MAFFT	T-Coffee	MUSCLE	TM-Coffee
Amino Acid	P92962						✓
	Q9SF09	✓		✓	✓		
	Q9ZU50	✓	✓	✓	✓	✓	✓
	Q84MA5	✓	✓	✓		✓	✓
	Q9FFL1	✓	✓	✓	✓	✓	✓
Phosphate	Q38954	✓	✓	✓	✓	✓	✓

Table 3.15: Unclassified sequences (First dataset).

Substrate	Sequence	ClustalW	Clustal Omega	MAFFT	T-Coffee	MUSCLE	TM -Coffee
Amino Acid	Q7TNK0	✓	✓	✓	✓	✓	✓
	Q8BLE7				✓		
	O35633				✓		
Anion	P39414		✓	✓	✓		
Cation	P60678	✓	✓	✓	✓	✓	✓
	P60698	✓	✓	✓	✓	✓	✓
	Q9NY26				✓		
	Q99JR1	✓	✓	✓	✓	✓	✓
	Q3TMP8	✓	✓	✓	✓	✓	✓
	Q9CPC8	✓		✓	✓	✓	✓
	P36606		✓		✓		
Electron	P0AAK4	✓	✓	✓	✓	✓	✓
	P09152	✓	✓	✓	✓	✓	✓
	P33599	✓	✓	✓	✓	✓	✓
	P0AFE0	✓	✓	✓		✓	✓
	P09193	✓	✓	✓	✓	✓	✓
	P0A616				✓		
Protein	P46970				✓		
	P40477	✓	✓	✓	✓		✓
	Q6URK4	✓	✓	✓	✓	✓	✓
	P52870	✓	✓	✓	✓	✓	✓
	P33754	✓	✓	✓	✓	✓	✓
	P0AG99	✓	✓	✓	✓	✓	✓
	P0ABU9		✓	✓	✓	✓	
	Q96QU8				✓		
	Q7Z412	✓	✓	✓	✓	✓	✓
Sugar	P39344	✓	✓	✓	✓	✓	✓
	O64503				✓		
	Q29Q28				✓		
	P71067	✓	✓	✓	✓	✓	✓
Other	P0AFF4	✓	✓	✓	✓	✓	✓
	O49929	✓	✓	✓	✓	✓	✓
	P38206	✓	✓	✓	✓	✓	
	Q9BZV2	✓	✓	✓	✓	✓	✓

Table 3.16: Unclassified sequences (Second dataset).

3.4.5 Impact of MSA ans SDS

Table 3.17, 3.18, 3.19 and 3.20 respectively show the impact of using different MSAs and the impact using three different SDS tools. From Table 3.17, we can see that comparing with the baseline which is using no MSA nor SDS, using different MSA tools most of the time has a positive impact. The best impact given is MASCLE, which $\Delta = 0.027$. The average impact Δ is 0.007. From Table 3.18, we can see that comparing with the baseline which is using no SDS, using Speer Server as SDS most of the time has a negative impact. The average impact Δ is -0.064. From Table 3.19, we can see that comparing with the baseline which is using no SDS, using GroupSim as SDS most of the time has a negative impact. The average impact Δ is -0.007. From Table 3.20, we can see that comparing with the baseline which is using no SDS, using Xdet as SDS has a positive impact. The average impact Δ is 0.001.

MSA	SDS	MCC	MSA	SDS	MCC	Δ
-	-	0.676	T-Coffee	-	0.636	-0.04
-	-	0.676	ClustalOmega	-	0.683	0.007
-	-	0.676	MAFFT	-	0.686	0.01
-	-	0.676	ClustalW	-	0.693	0.017
-	-	0.676	TM-Coffee	-	0.697	0.021
-	-	0.676	MUSCLE	-	0.703	0.027
					Average	0.007

Table 3.17: Impact of different MSA tools with baseline (Second dataset).
The results come from the independent test set of size 120. The third column "MCC" is the MCC for baseline, which uses no MSA nor SDS. The sixth column "MCC" is the MCC for baselines, which use different MSAs and no SDS. The seventh column "Δ" is means the differences between MSA baseline and no MSA baseline. This table is sorted by ascending of the sixth column "MCC".

MSA	SDS	MCC	MSA	SDS	MCC	Δ
T-Coffee	-	0.636	T-Coffee	Speer Server	0.639	0.003
ClustalOmega	-	0.683	ClustalOmega	Speer Server	0.593	-0.07
MAFFT	-	0.686	MAFFT	Speer Server	0.591	-0.095
ClustalW	-	0.693	ClustalW	Speer Server	0.612	-0.018
TM-Coffee	-	0.697	TM-Coffee	Speer Server	0.605	-0.092
MUSCLE	-	0.703	MUSCLE	Speer Server	0.593	-0.11
					Average	-0.064

Table 3.18: Impact of different MSA tools with baseline (Second dataset).
The results come from the independent test set of size 120. The third column "MCC" is the MCC for baselines, which use different MSAs and no SDS. The sixth column "MCC" is the MCC for using different MSAs and Speer Server as SDS. The seventh column "Δ" is means the differences between no SDS baseline and using Speer Server as SDS. This table is sorted by ascending of the third column "MCC".

MSA	SDS	MCC	MSA	SDS	MCC	Δ
T-Coffee	-	0.636	T-Coffee	GroupSim	0.638	0.002
ClustalOmega	-	0.683	ClustalOmega	GroupSim	0.688	0.005
MAFFT	-	0.686	MAFFT	GroupSim	0.668	-0.018
ClustalW	-	0.693	ClustalW	GroupSim	0.685	-0.008
TM-Coffee	-	0.697	TM-Coffee	GroupSim	0.689	-0.008
MUSCLE	-	0.703	MUSCLE	GroupSim	0.690	-0.013
					Average	-0.007

Table 3.19: Impact of different MSA tools with baseline (Second dataset).
The results come from the independent test set of size 120. The third column "MCC" is the MCC for baselines, which use different MSAs and no SDS. The sixth column "MCC" is the MCC for using different MSAs and GroupSim as SDS. The seventh column "Δ" is means the differences between no SDS baseline and using GroupSim as SDS. This table is sorted by ascending of the third column "MCC".

MSA	SDS	MCC	MSA	SDS	MCC	Δ
T-Coffee	-	0.636	T-Coffee	Xdet	0.638	0.002
ClustalOmega	-	0.683	ClustalOmega	Xdet	0.681	-0.002
MAFFT	-	0.686	MAFFT	Xdet	0.702	0.016
ClustalW	-	0.693	ClustalW	Xdet	0.692	-0.001
TM-Coffee	-	0.697	TM-Coffee	Xdet	0.675	-0.022
MUSCLE	-	0.703	MUSCLE	Xdet	0.716	0.013
					Average	0.001

Table 3.20: Impact of different MSA tools with baseline (Second dataset).
The results come from the independent test set of size 120. The third column "MCC" is the MCC for baselines, which use different MSAs and no SDS. The sixth column "MCC" is the MCC for using different MSAs and Xdet as SDS. The seventh column "Δ" is means the differences between no SDS baseline and using Xdet as SDS. This table is sorted by ascending of the third column "MCC".

Chapter 4

Conclusion

In this thesis we have investigated using profile Hidden Markov Models and specificity determining sites to classify the substrate class of a transmembrane transport protein. We followed the evaluation methodology and used the same dataset of two earlier works on the topic, namely the small dataset and methodology of the Helms lab [44], and the large dataset and methodology of the Zhao lab [33]. In the end, our approach led to a combination of the MUSCLE MSA tool and the XDet SDS tool that achieved superior MCC to the state-of-the-art system, TrSSP [].

This chapter will present our contributions in Section 4.1, the limitation of our work in Section 4.2, and then Section 4.3 provides recommendations for the future work.

4.1 Contributions

The contributions of this thesis are as follows:

(1) This work is the first to apply Hidden Markov Models to classify the substrate class of a transmembrane transport protein, and the first to combine the specificity determining sites with profile Hidden Markov Model to the classification for the substrate class transported across a membrane by a given transmembrane transporter protein.

(2) We have established a classification pipeline which integrated the steps of data pre-processing, model building and model evaluation. The pipeline contains similarity search, multiple sequence alignment, specificity determining site prediction and construction of a Hidden Markov

Model.

(3) We discuss and analyze the impact of the using different MSA tools, SDS tools and parameters on the classification of the substrate class. We find a combination of MUSCLE and Xdet which outperforms the state-of-the-art TrSSP.

4.2 Limitations of the Work

The small dataset [44] has only 61 sequences. It did not provide enough examples to see differences between the combinations of tools in the pipeline. The large dataset with 780 sequences in the training set and 120 sequences in the test set was much better for comparing combinations of tools in the pipeline.

The literature has a large number of tools, methods, algorithms for both multiple sequence alignment (MSA) and prediction of specificity determining sites (SDS). We did not try all of them. For MSA, we focused on the most widely used tools, but could have tried more. For SDS, many of the tools in the literature are not available: we used the ones that we could find software to install and work on our systems. There may be others that we did not try.

The small dataset covers only four substrate classes. The large datatset contains examples from seven substrate classes, one of which is *"Other"*. In the literature, when scientists apply the TCDB database to annotate transmembrane transport proteins in a genome, they typically distinguish 27 substrate classes. Hence, our field should develop larger datatsets with coverage of more substrate classes.

classifying the substrate class of a transmembrane transport protein is important, but on many occasions scientists want more detailed information. They would like to know the **specific** substrate, not the **class** of the substrate, that is actually transported. At the moment, we do not have enough examples where the specific substrate is known, in order to attempt the classification of the specific substrate.

4.3 Future Work

The limitations noted in the last section call attention to several areas that we deem worthy of further investigation.

- Extend the dataset in size and in coverage of the substrate classes.
- Apply the pipeline with other classifiers beyond Hidden Markov Models, such as Support Vector Machines.
- classify the specific substrate (e.g. Lysine) rather than the substrate class (e.g. amino acid), at least for those cases where there is sufficent data available.

Appendix A

The First Dataset

Uniprot ID	TCDB Family
P92934	2.A.18 The amino acid/auxin permease (AAAP)
P92961	2.A.18 The amino acid/auxin permease (AAAP)
P92962	2.A.18 The amino acid/auxin permease (AAAP)
Q38967	2.A.18 The amino acid/auxin permease (AAAP)
Q39134	2.A.18 The amino acid/auxin permease (AAAP)
Q42400	2.A.18 The amino acid/auxin permease (AAAP)
Q8GUM3	2.A.18 The amino acid/auxin permease (AAAP)
Q9FN04	2.A.18 The amino acid/auxin permease (AAAP)
Q9SF09	2.A.18 The amino acid/auxin permease (AAAP)
Q9SJP9	2.A.18 The amino acid/auxin permease (AAAP)
Q9ZU50	2.A.3 The amino acid-polyamine-organocation (APC)
Q84MA5	2.A.3 The amino acid-polyamine-organocation (APC)
Q9FFL1	2.A.3 The amino acid-polyamine-organocation (APC)
Q8W4K3	2.A.3 The amino acid-polyamine-organocation (APC)

Table A.1: Amino Acid

Uniprot ID	TCDB Family
O04036	2.A.1.1 The sugar porter (SP)
P23586	2.A.1.1 The sugar porter (SP)
Q0WWW9	2.A.1.1 The sugar porter (SP)
Q39228	2.A.1.1 The sugar porter (SP)
Q56ZZ7	2.A.1.1 The sugar porter (SP)
Q8L6Z8	2.A.1.1 The sugar porter (SP)
Q9C757	2.A.1.1 The sugar porter (SP)
Q9FMX3	2.A.1.1 The sugar porter (SP)
Q2V4B9	2.A.1.1 The sugar porter (SP)
Q6AWX0	2.A.1.1 The sugar porter (SP)
Q8GW61	2.A.1.1 The sugar porter (SP)
Q8GXR2	2.A.1.1 The sugar porter (SP)
Q9LNV3	2.A.123 The sweet; PQ-loop; saliva; MtN3 (Sweet)
Q8L9J7	2.A.123 The sweet; PQ-loop; saliva; MtN3 (Sweet)
Q9SMM5	2.A.123 The sweet; PQ-loop; saliva; MtN3 (Sweet)

Table A.2: Hexose

Uniprot ID	TCDB Family
O04514	2.A.67 The oligopeptide transporter (OPT)
O23482	2.A.67 The oligopeptide transporter (OPT)
O82485	2.A.67 The oligopeptide transporter (OPT)
Q9FG72	2.A.67 The oligopeptide transporter (OPT)
Q9FJD1	2.A.67 The oligopeptide transporter (OPT)
Q9FME8	2.A.67 The oligopeptide transporter (OPT)
Q9SUA4	2.A.67 The oligopeptide transporter (OPT)
Q9T095	2.A.67 The oligopeptide transporter (OPT)
Q9FJD2	2.A.67 The oligopeptide transporter (OPT)
P46032	2.A.67 The oligopeptide transporter (OPT)
Q05085	2.A.17 The oligopeptide transporter (OPT)
Q9LFX9	2.A.17 The proton-dependent oligopeptide transporter (POT/PTR)
Q9LSE8	2.A.17 The proton-dependent oligopeptide transporter (POT/PTR)
Q9M172	2.A.17 The proton-dependent oligopeptide transporter (POT/PTR)
Q9M174	2.A.17 The proton-dependent oligopeptide transporter (POT/PTR)
Q9M175	2.A.17 The proton-dependent oligopeptide transporter (POT/PTR)
Q9SZY4	2.A.17 The proton-dependent oligopeptide transporter (POT/PTR)

Table A.3: Oligopeptide

Uniprot ID	TCDB Family
O48639	2.A.1.9 The phosphate: H+ symporter (PHS)
Q8VYM2	2.A.1.9 The phosphate: H+ symporter (PHS)
Q96243	2.A.1.9 The phosphate: H+ symporter (PHS)
Q9S735	2.A.1.9 The phosphate: H+ symporter (PHS)
Q96303	2.A.1.14 The anion:cation symporter (ACS)
Q9FKV1	2.A.1.14 The anion:cation symporter (ACS)
O82390	2.A.1.14 The anion:cation symporter (ACS)
Q3E9A0	2.A.1.14 The anion:cation symporter (ACS)
Q38954	2.A.20 The inorganic phosphate transporter (PiT)
Q7DNC3	2.A.29 The mitochondrial carrier (MC)
Q9FMU6	2.A.29 The mitochondrial carrier (MC)
Q9M2Z8	2.A.29 The mitochondrial carrier (MC)
Q8H0T6	2.A.7 The drug/metabolite transporter (DMT)
Q8RXN3	2.A.7 The drug/metabolite transporter (DMT)
Q94B38	2.A.7 The drug/metabolite transporter (DMT)

Table A.4: Phosphate

Appendix B

The Second Dataset

Uniprot ID	TCDB Family
B0R9X2	2.A.35.1.4 the nhac Na(+):H(+) antiporter (NHAC) family
P38084	2.A.3.10.6 the amino acid-polyamine-organocation (APC) family
P15365	2.A.1.14.4 the major facilitator superfamily (MFS)
P36837	2.A.17.1.3 the proton-dependent oligopeptide transporter (POT/PTR) family
Q9Z127	
Q8TF71	2.A.1.13.8 the major facilitator superfamily (MFS)
Q06593	2.A.67.1.4 the oligopeptide transporter (OPT) family
O01840	2.A.17.4.10 the proton-dependent oligopeptide transporter (POT/PTR) family
Q6YBV0	2.A.18.8.5 the amino acid/auxin permease (AAAP) family
Q9H2J7	2.A.22.6.7 the neurotransmitter:sodium symporter (NSS) family
Q7TNK0	
P38967	2.A.3.10.8 the amino acid-polyamine-organocation (APC) family
Q8BLE7	
Q10901	2.A.23.2.10 the dicarboxylate/amino acid:cation (Na(+) or H(+)) symporter (DAACS) family
O35633	2.A.18.5.3 the amino acid/auxin permease (AAAP) family

Table B.1: Amino Acid.

Uniprot ID	TCDB Family
P51788	2.A.49.2.12 the chloride carrier/channel (CLC) family
P21439	3.A.1.201.3 the ATP-binding cassette (ABC) superfamily
A0QQ70	3.A.1.9.2 the ATP-binding cassette (ABC) superfamily
P39535	2.A.47.2.3 the divalent anion: Na(+) symporter (DASS) family
Q8RX77	2.A.17.3.15 the proton-dependent oligopeptide transporter (pot/ptr) family
Q80UJ1	2.A.1.19.16 the major facilitator superfamily (MFS)
Q5DTL9	2.A.31.2.3 the anion exchanger (AE) family
Q9NPD5	2.A.60.1.12 the organo anion transporter (OAT) family
P39414	2.A.47.3.3 the divalent anion: Na(+) symporter (DASS) family
P37020	2.A.49.1.1 the chloride carrier/channel (CLC) family
Q02785	3.A.1.205.3 the ATP-binding cassette (ABC) superfamily
P44543	2.A.56.1.3 the tripartite ATP-independent periplasmic transporter (TRAP-t) family

Table B.2: Anion.

Uniprot ID	TCDB Family
Q8AYS8	R-GGA-1296052 Ca2+ activated K+ channels R-RNO-1296072 Voltage gated Potassium channels
Q02280	1.A.1.20.6 the voltage-gated ion channel (VIC) superfamily
Q03721	
Q9UJ96	
O00180	
Q9ES08	
P51787	1.A.1.15.6 the voltage-gated ion channel (VIC) superfamily
Q6UVM4	
Q9YDF8	1.A.1.17.1 the voltage-gated ion channel (VIC) superfamily
P40260	1.A.11.3.1 the ammonium transporter channel (amt) family
P60678	2.A.63.1.3 the monovalent cation (K(+) or Na(+)):proton antiporter-3 (CPA3) family
P60698	2.A.63.1.3 the monovalent cation (k(+) or NA(+)):proton antiporter-3 (CPA3) family
Q9ZT63	2.A.4.3.4 the cation diffusion facilitator (CDF) family
P13738	2.A.33.1.1 the nhaa NA(+):H(+) antiporter (NHAA) family
Q68KI4	2.A.36.5.1 the monovalent cation:proton antiporter-1 (CPA1) family
Q8BHK1	
Q8N130	2.A.58.1.3 the phosphate:NA(+) symporter (PNAS) family
Q96SN7	
Q10177	2.A.55.1.4 the metal ion (mn(2+)-iron) transporter (NRAMP) family
Q8BUX5	
Q9BXP2	2.A.30.1.6 the cation-chloride cotransporter (CCC) family
Q9NY26	2.A.5.3.2 the zinc (zn(2+))-iron (fe(2+)) permease (ZIP) family
P04775	
Q62968	1.A.1.10.6 the voltage-gated ion channel (VIC) superfamily
Q99JR1	
Q96T83	2.A.36.1.3 the monovalent cation:proton antiporter-1 (CPA1) family
Q3TMP8	1.A.62.1.1 the homotrimeric cation channel (TRIC) family
P28584	2.A.38.2.3 the K(+) transporter (TRK) family
P42839	2.A.19.7.1 the Ca(2+):cation antiporter (CACA) family
Q8IUH5	9.B.37.1.1 the huntington-interacting protein 14 (HIP14) family
P34240	2.A.5.5.3 the zinc (Zn(2+))-iron (Fe(2+)) permease (ZIP) family
P19657	
P32842	3.A.2.2.3 the H(+)- or NA(+)-translocating f-type, v-type and a-type ATPase (f-ATPase) superfamily
Q9CPC8	
P36606	2.A.36.1.21 the monovalent cation:proton antiporter-1 (CPA1) family 2.A.36.4.3 the monovalent cation:proton antiporter-1 (CPA1) family

Table B.3: Cation.

Uniprot ID	TCDB Family
Q97K92	
P0A8Q0	
P0AAK4	
P09152	5.A.3.1.1 the prokaryotic molybdopterin-containing oxidoreductase (pmo) family
Q00141	
P33599	3.D.1.1.1 the H(+) or NA(+)-translocating NADH dehydrogenase (ndh) family
P0AFE0	3.D.1.1.1 the H(+) or NA(+)-translocating NADH dehydrogenase (ndh) family
P09193	3.E.2.2.2 the photosynthetic reaction center (prc) family
P0A616	
Q97D82	

Table B.4: Electron transporter

Uniprot ID	TCDB Family
P0ADC3	3.A.1.125.1 the ATP-binding cassette (abc) superfamily
P46970	
P40477	1.I.1.1.1 the eukaryotic nuclear pore complex (e-npc) family
Q6URK4	
P52870	3.A.5.8.1 the general secretory pathway (sec) family
P33754	3.A.5.8.1 the general secretory pathway (sec) family
P0AG99	3.A.5.1.1 the general secretory pathway (sec) family
P69428	2.A.64.1.1 the twin arginine targeting (tat) family
P32830	
P39515	3.A.8.1.1 the mitochondrial protein translocase (mpt) family
Q9Y5J7	
P0ABU9	1.A.30.2.2 the H(+)- or NA(+)-translocating bacterial flagellar motor/exbbd outer membrane transport energizer (mot/exb) superfamily
Q96QU8	
P33331	
Q7Z412	3.A.20.1.1 the peroxisomal protein importer (ppi) family

Table B.5: Protein/mRNA.

Uniprot ID	TCDB Family
Q9JIF3	2.A.1.1.46 the major facilitator superfamily (mfs)
P39344	2.A.8.1.2 the gluconate:H(+) symporter (gntp) family
P68183	3.A.1.1.1 the ATP-binding cassette (abc) superfamily
Q57071	4.A.1.1.13 the pts glucose-glucoside (glc) family
P33026	2.A.1.20.2 the major facilitator superfamily (mfs)
Q94AZ2	2.A.1.1.50 the major facilitator superfamily (mfs)
O80605	2.A.2.4.3 the glycoside-pentoside-hexuronide (gph): cation symporter family
A1Z8N1	2.A.1.1.99 the major facilitator superfamily (mfs)
P0AGC0	2.A.1.4.1 the major facilitator superfamily (mfs)
O64503	2.A.7.11.4 the drug/metabolite transporter (dmt) superfamily
Q29Q28	
P71067	2.A.14.1.3 the lactate permease (lctp) family

Table B.6: Sugar.

Uniprot ID	TCDB Family
P0AFF4	2.A.1.10.1 the major facilitator superfamily (mfs)
O49929	1.B.28.1.1 the plastid outer envelope porin of 24 kda (oep24) family
P53860	
Q9LU77	2.A.69.1.2 the auxin efflux carrier (aec) family
P43286	1.A.8.11.4 the major intrinsic protein (mip) family
P69874	3.A.1.11.1 the ATP-binding cassette (abc) superfamily
Q04671	2.A.45.2.1 the arsenite-antimonite (arsb) efflux family
P38206	2.A.66.3.1 the multidrug/oligosaccharidyl-lipid/polysaccharide (mop) flippase superfamily
Q9BZV2	2.A.48.1.4 the reduced folate carrier (rfc) family
Q60714	
Q14542	2.A.57.1.8 the equilibrative nucleoside transporter (ent) family
Q9Y345	2.A.22.2.10 the neurotransmitter:sodium symporter (nss) family
D0ZXQ3	
Q41963	1.A.8.10.9 the major intrinsic protein (mip) family
P53099	2.A.39.2.2 the nucleobase:cation symporter-1 (ncs1) family
P04633	
Q8VHL0	
Q60932	
P67444	2.A.40.4.3 the nucleobase/ascorbate transporter (nat) or nucleobase:cation symporter-2 (ncs2) family
Q12675	3.A.3.8.5 the p-type ATPase (p-ATPase) superfamily

Table B.7: Other transporter.

Bibliography

[1] Saraswathi Abhiman and Erik LL Sonnhammer. Large-scale prediction of function shift in protein families with a focus on enzymatic function. *Proteins: Structure, Function, and Bioinformatics*, 60(4):758–768, 2005.

[2] SF Altschul, W Gish, W Miller, EW Myers, and DJ Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[3] SF Altschul, TL Madden, AA Schäffer, J Zhang, Z Zhang, W Miller, and DJ Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[4] F. Aplop and G. Butler. On predicting transport proteins and their substrates for the reconstruction of metabolic networks. In *2015 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–9, 2015.

[5] Amos Bairoch. The ENZYME database in 2000. *Nucleic Acids Research*, 28(1):304–305, 2000.

[6] Mohamed Bekkar, Hassiba Kheliouane Djemaa, and Taklit Akrouf Alitouche. Evaluation measures for models assessment over imbalanced data sets. *Journal of Information Engineering and Applications*, 3(10), 2013.

[7] Gordon Blackshields, Fabian Sievers, Weifeng Shi, Andreas Wilm, and Desmond G. Higgins. Sequence embedding for fast construction of guide trees for multiple sequence alignment. *Algorithms for Molecular Biology*, 5(1):21, 2010.

[8] Blazej Bulka, Marie desJardins, and Stephen J. Freeland. An interactive visualization tool to explore the biophysical properties of amino acids and their contribution to substitution matrices. *BMC Bioinformatics*, 7(1):329, Jul 2006.

[9] Ahmad Hassan Butt, Nouman Rasool, and Yaser Daanial Khan. A treatise to computational approaches towards prediction of membrane protein and its subtypes. *The Journal of Membrane Biology*, 250(1):55–76, 2017.

[10] James C Whisstock and Arthur M Lesk. Prediction of protein function from protein sequence and structure. *Quarterly Reviews of Biophysics*, 36:307–40, 09 2003.

[11] John A. Capra and Mona Singh. Characterization and prediction of residues determining protein functional specificity. *Bioinformatics*, 24(13):1473–1480, 2008.

[12] Elisabeth P Carpenter, Konstantinos Beis, Alexander D Cameron, and So Iwata. Overcoming the challenges of membrane protein crystallography. *Current Opinion in Structural Biology*, 18(5):581–586, 2008.

[13] Saikat Chakrabarti, Stephen H. Bryant, and Anna R. Panchenko. Functional specificity lies within the properties and evolutionary changes of amino acids. *Journal of Molecular Biology*, 373(3):801 – 810, 2007.

[14] Abhijit Chakraborty and Saikat Chakrabarti. A survey on prediction of specificity-determining sites in proteins. *Briefings in Bioinformatics*, 16(1):71–88, 2015.

[15] Kuo-Chen Chou. Prediction of protein subcellular locations by incorporating quasi-sequence-order effect. *Biochemical and Biophysical Research Communications*, 278(2):477–483, 2000.

[16] Zejin Ding. *Diversified ensemble classifiers for highly imbalanced data learning and their application in bioinformatics*. PhD thesis, Georgia State University, 2011.

[17] Inna Dubchak, Ilya Muchnik, Christopher Mayor, Igor Dralyuk, and Sung-Hou Kim. Recognition of a protein fold in the context of the Structural Classification Of Proteins (SCOP) classification. *Proteins: Structure, Function, and Bioinformatics*, 35(4):401–407, 1999.

[18] S R Eddy. Profile hidden markov models. *Bioinformatics*, 14(9):755–763, 1998.

[19] Sean Eddy. HMMER: biosequence analysis using profile hidden markov models, 2014.

[20] Robert C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797, 2004.

[21] Robert C Edgar and Serafim Batzoglou. Multiple sequence alignment. *Current Opinion in Structural Biology*, 16(3):368 – 373, 2006.

[22] Evan W. Floden, Paolo D. Tommaso, Maria Chatzou, Cedrik Magis, Cedric Notredame, and Jia-Ming Chang. PSI/TM-Coffee: a web server for fast and accurate multiple sequence alignments of regular and transmembrane proteins using homology extension on reduced databases. *Nucleic Acids Research*, 44(W1):W339–W343, 2016.

[23] M. Michael Gromiha and Yukimitsu Yabuki. Functional discrimination of membrane proteins using machine learning techniques. *BMC Bioinformatics*, 9(1):135, 2008.

[24] Xun Gu. Maximum-likelihood approach for gene family evolution under functional divergence. *Molecular Biology and Evolution*, 18(4):453–464, 2001.

[25] S Henikoff and J G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[26] Xiaoqiu Huang and Webb Miller. A time-efficient, linear-space local similarity algorithm. *Advances in Applied Mathematics*, 12(3):337 – 357, 1991.

[27] K. Katoh, K. Misawa, K. i. Kuma, and T. Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Research*, 30(14):3059–3066, 2002.

[28] Motoo Kimura. The neutral theory of molecular evolution: A review of recent evidence. *The Japanese Journal of Genetics*, 66(4):367–386, 1991.

[29] Dániel Kozma, István Simon, and Gábor E. Tusnády. PDBTM: Protein data bank of transmembrane proteins after 8 years. *Nucleic Acids Research*, 41(D1):D524–D529, 2013.

[30] Anders Krogh, Michael Brown, I.Saira Mian, Kimmen Sjölander, and David Haussler. Hidden markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235(5):1501 – 1531, 1994.

[31] Haiquan Li, Vagner A. Benedito, Michael K. Udvardi, and Patrick Xuechun Zhao. TransportTP: A two-phase classification approach for membrane transporter prediction and characterization. *BMC Bioinformatics*, 10(1):418, 2009.

[32] Harvey Lodish, Arnold Berk, James E Darnell, Chris A Kaiser, Monty Krieger, Matthew P Scott, Anthony Bretscher, Hidde Ploegh, and Paul Matsudaira. Genes, Genomics, and Chromosomes. In *Molecular Cell Biology*, chapter 6. Macmillan, 2008.

[33] Nitish K. Mishra, Junil Chang, and Patrick X. Zhao. Prediction of membrane transport proteins and their substrate specificities using primary sequence information. *PLOS ONE*, 9(6):1–14, 06 2014.

[34] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453, 1970.

[35] Cédric Notredame, Desmond G Higgins, and Jaap Heringa. T-Coffee: a novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302(1):205 – 217, 2000.

[36] Florencio Pazos, Antonio Rausell, and Alfonso Valencia. Phylogeny-independent detection of functional residues. *Bioinformatics*, 22(12):1440–1448, 2006.

[37] Alexander Pertsemlidis and John W Fondon. Having a blast with bioinformatics (and avoiding blastphemy). *Genome Biol*, 2(10), 2001.

[38] Jonathan Pevsner. Multiple sequence alignment. In *Bioinformatics and Functional Genomics*, chapter 6. Wiley-Blackwell, 2009.

[39] Walter Pirovano, K. Anton Feenstra, and Jaap Heringa. Praline: a strategy for improved multiple alignment of transmembrane proteins. *Bioinformatics*, 24(4):492–497, 2008.

[40] Tal Pupko, Rachel Bell, Itay Mayrose, Fabian Glaser, and Nir Ben-Tal. Rate4site: An algorithmic tool for the identification of functional regions in proteins by surface mapping of evolutionary determinants within their homologues. *Bioinformatics (Oxford, England)*, 18 Suppl 1:S71–7, 02 2002.

[41] N Saitou and M Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.

[42] F. Sanger. The arrangement of amino acids in proteins. In M.L. Anson, Kenneth Bailey, and John T. Edsall, editors, *Advances in Protein Chemistry*, volume 7, pages 1 – 67. Academic Press, 1952.

[43] N. S. Schaadt and V. Helms. Functional classification of membrane transporters and channels based on filtered TM/nonTM amino acid composition. *Biopolymers*, 97(7):558–567, 7 2012.

[44] Nadine S. Schaadt, Jan Christoph, and Volkhard Helms. Classifying substrate specificities of membrane transporters from *Arabidopsis thaliana*. *Journal of Chemical Information and Modeling*, 50(10):1899–1905, 2010.

[45] Johannes Söding. Protein homology detection by HMMHMM comparison. *Bioinformatics*, 21(7):951–960, 2005.

[46] J-Y Shi, S-W Zhang, Q Pan, Y-M Cheng, and J Xie. Prediction of protein subcellular localization by support vector machines using multi-scale energy and pseudo amino acid composition. *Amino Acids*, 33(1):69–74, 2007.

[47] Fabian Sievers, Andreas Wilm, David Dineen, Toby J Gibson, Kevin Karplus, Weizhong Li, Rodrigo Lopez, Hamish McWilliam, Michael Remmert, Johannes Söding, Julie D Thompson, and Desmond G Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, 7(1), 2011.

[48] Kimmen Sjölander. Phylogenetic inference in protein superfamilies: Analysis of SH2 domains. *International Conference on Intelligent Systems for Molecular Biology*, 6:165–74, 02 1998.

[49] J D Thompson, D G Higgins, and J Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):46734680, 1994.

[50] Gary M Weiss and Foster Provost. Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–354, 2003.