# *TPORTHMM*: PREDICTING THE SUBSTRATE CLASS OF TRANSMEMBRANE TRANSPORT PROTEINS USING PROFILE HIDDEN MARKOV MODELS

Shiva Shamloo

A thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

December 2020

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

|  |  |
|---|---|
| By: | **Shiva Shamloo** |
| Entitled: | ***TportHMM*: Predicting the substrate class of transmembrane transport proteins using profile Hidden Markov Models** |

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

**Master of Computer Science**

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

Signed by the final examining commitee:

______________________________ Examiner
Dr. Sabine Bergler

______________________________ Examiner
Dr. Andrew Delong

______________________________ Supervisor
Dr. Gregory Butler

Approved ______________________________

Dr. Lata Narayanan, Chair

Department of Computer Science and Software Engineering

__________ 20 ______ ______________________________

Dean Dr. Mourad Debbabi

Faculty of Engineering and Computer Science

# Abstract

*TportHMM*: Predicting the substrate class of transmembrane transport proteins using profile Hidden Markov Models

Shiva Shamloo

Transporters make up a large proportion of proteins in a cell, and play important roles in metabolism, regulation, and signal transduction by mediating movement of compounds across membranes but they are among the least characterized proteins due to their hydrophobic surfaces and lack of conformational stability. There is a need for tools that predict the substrates which are transported at the level of substrate class and the level of specific substrate.

This work develops a predictor, *TportHMM*, using profile Hidden Markov Model (HMM) and Multiple Sequence Alignment (MSA). We explore the role of multiple sequence alignment (MSA) algorithms to utilise evolutionary information, specificity-determining site (SDS) algorithms to highlight positional information, and a profile Hidden Markov Model (HMM) classifier to utilise sequence information.

We study the impact of different MSA algorithms (ClustalW, Clustal Omega, MAFFT, MUSCLE, AQUA, T-Coffee and TM-Coffee), and different SDS algorithms (Speer Server, GroupSim, Xdet and TCS). We compare these approaches with the state-of-the-art, *TrSSP* and *TranCEP*.

# Acknowledgments

I am really fortunate that I had the great supervision of Dr. Gregory Butler. His exemplary guidance, carefull monitoring and amicable support throughout my master's are so great that even my most profound gratitude is not enough.

I cannot express enough thanks to my amazing mother for her unconditional love and encouragement during my life. There are not enough words to describe her patience and I certainly could not have been here with her support. My sincere thanks to my father for his aid in teaching me how to think mathematically.

My deepest gratitute goes to Ebi joon, for his kindness, warm-hearted and tolerant patronage during my studies. His advisory has been one of the principal motivations in my life. Finally, I would also thank my grandmother, my aunts and uncles for their infinite reliance and kindness.

# 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

**Secator** A program for inferring protein subfamilies from phylogenetic trees

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

**SDS** Specificity determining site

**GroupSim** 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`

**TCS** A program for evaluating alignment using transitive consistency score

**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 multiple predictors, using profile Hidden Markov Model (HMM) for classifying the substrates that are transported across a membrane by a transmembrane transport protein.

These predictors are assembled as the pipeline of multiple sequence alignment (MSA), prediction of specificity determining sites (SDS), and the HMMER tool `hmmbuild`.

In this chapter, we include a brief summary on the basic biological knowledge of membrane and transmembrane proteins in Section 1.1 ; 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

A single cell consists of three parts: the cell membrane, the nucleus, and, between the two, the cytoplasm. The cell membrane separates the material outside the cell, extracellular, from the material inside the cell, intracellular. It maintains the integrity of a cell and controls passage of materials into and out of the cell.

Cell membranes consist of two main components: lipids and proteins as shown in Figure 1. Membrane proteins are important proteins for organisms. They enable the membrane to perform distinctive activities with a vast diversity of cell membrane functions. Membrane proteins are classified into three different groups: integral membrane proteins (IMP), lipid-anchored membrane proteins and peripheral membrane proteins as shown in Figure 1. This work focuses on the integral membrane proteins, which are also called transmembrane proteins. In particular, we are concerned with the transmembrane transport proteins.

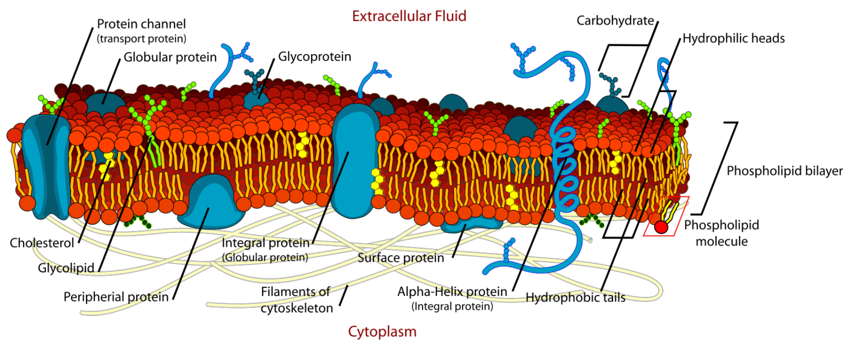


Figure 1: The cell membrane

The cell membrane separates the inside and outside of the cell. It contains a variety of biological molecules, mailnly proteins and lipids. [Com20b]. [Fla19].

Transporters are transmembrane proteins involved with the movement of ions, small molecules, and macromolecules across the inner and outer membranes of a cell. Experimental characterization of their structure and function is exceptionally difficult, due to their hydrophobic surfaces and their lack of conformational stability. An example of this limitation is the Protein Data Bank (PDB). As of March 2020, less than 4% of the `PDB` is membrane proteins. Hence, there is a need for computational approaches to distinguish and characterize the substrates that they transport.

## 1.2 Related Works

For most of the work done on the prediction of transport proteins [GO14], there is no available software, so it is difficult to reproduce the work and to compare the results of different articles. *TranCEP* [AAB20] is the state-of-the-art, surpassing *TrSSP* [MCZ14]. Both these articles discuss the previous work in detail.

The *TranCEP* system is based on a Support Vector Machine (SVM) classifier using amino acid composition (AAC) vectors. The work explored several methods of AAC with TM-Coffee as the single choice of MSA and TCS as the single choice of SDS. The best combination was TM-Coffee with TCS and pairwise amino acid composition (PAAC).

The *TrSSP* system uses a Support Vector Machine (SVM) with input from a profile PSSM and the AAIndex physicochemical composition and varies it to illustrate the impact of each of the factors: compositional, evolutionary, and positional information. This work investigate profile HMM classifiers rather than SVM, and investigate a broad range of MSA and SDS tools in the pipeline.

Multiple sequence alignments algorithms are surveyed in [Not07, Not02] and compared through benchmarking in [TPP99]. The MSA methods used for this experiment are ClustalW [THG94], MAFFT [KMKM02], MUSCLE [Edg04], TM-Coffee [FTC+16], AQUA [MCT+09], ClustalOmega [SWD+11], and T-Coffee [NHH00].

Positional information in terms of the amino acid location within the sequence, as determined by analysing an MSA for "important" positions (columns), whether based on conservation, mutual information, or specificity determining positions [TWNB12]. The available methods for these specificity-determining sites are surveyed in [CC15]. The SDS methods used are Speer Server [CML+12], GroupSim [CS08], Xdet [PRV06], and TCS [CDTN14].

## 1.3 Research Contribution

The contributions of this thesis are as follows:

1. This work is the first to apply Hidden Markov Models for classification of transporter protein substrates and the first to utilise the combination of specificity determining sites and Hidden Markov Model to classify the substrate class of a transmembrane transport protein based on a given transmembrane transporter protein.

2. We establish a pipeline for classification of the membrane proteins which integrates the steps of data processing, model building and model evaluation. It consists of similarity search, multiple sequence alignment, specificity determining site prediction

and construction of a profile Hidden Markov Model. We perform extensive testing and analysis of different combinations of MSA and SDS tools for two different datasets.

3. We study the impact of different MSA algorithms (ClustalW, Clustal Omega, MAFFT, MUSCLE, AQUA, T-Coffee and TM-Coffee), and different SDS algorithms (Speer Server, GroupSim, Xdet and TCS). We compare these approaches with the state-of-the-art, *TrSSP* and *TranCEP*.

## 1.4 Thesis Outline

The remainder of this thesis is organized as follows:

**Chapter 2** introduces the background necessary for understanding the work in this thesis. Section 2.1 describes the structure of the proteins as well as the the function of the different types on integral membrane proteins. Section 2.2 and Section 2.3 present the algorithms for proteins sequence analysis, Section 2.4 outline a protein database, Section 2.5 to Section 2.8 describes the tools used in this thesis.

**Chapter 3** explains our pipeline implementation in detail. Section 3.1 presents the datasets and database used in this project. Section 3.2 clarifies each method used in our implementation in detail. Section 3.3 respectively shows the experimental results and discussion on our work.

**Chapter 4** concludes the thesis. Section 4.1 indicates some of the limitations that we faced in this experiment. Section 4.2 suggests some directions worthy of further investigation.

# Chapter 2

# Background

## 2.1 Membrane Proteins

### 2.1.1 Proteins

Proteins are built up from the blocks of alpha ($\alpha$) amino acids. Thus, proteins are represented by a sequence of amino acids. The structure of amino acid is shown in Figure 2. Amino acids have 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 which we call R-group as shown. They are divided into seveveral categories by their : side chain, size, shape, charge and hydrophobicity. There are 20 different amino acids that appear in the genetic code. They can be represented by English letters as shown in Table 1. An amino acid sequence helps determining the corresponding protein's 3D structure and by knowing the 3D structure, one can understand the protein functionality. The linear sequence of amino acids is called the protein primary structure.

### 2.1.2 Integral Membrane proteins

As mentioned in Section 1.1, the cell membrane separates the inside from the outside of the cell. Membrane proteins consist of three main categories: integral proteins, peripheral proteins, and lipid-anchored proteins. The integral membrane proteins transport specific ions, sugars, amino acids, and vitamins to cross the impermeable phospholipid bilayer into the cell

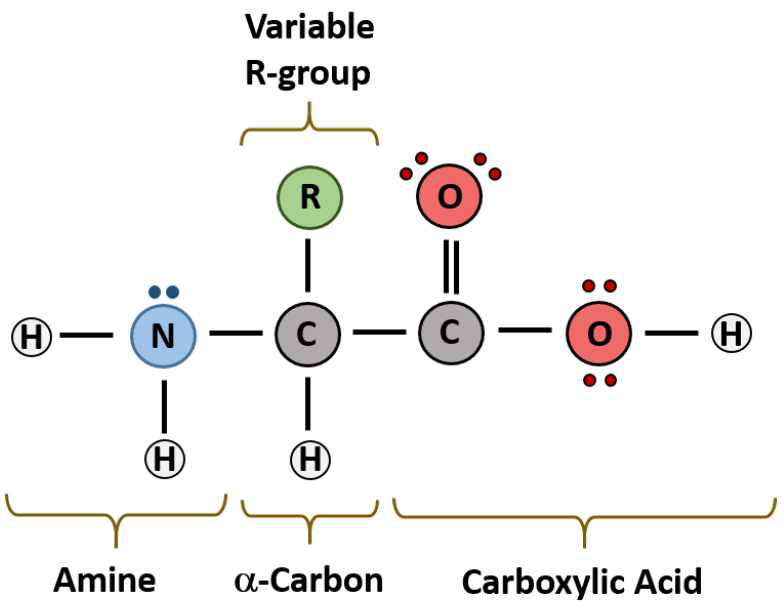
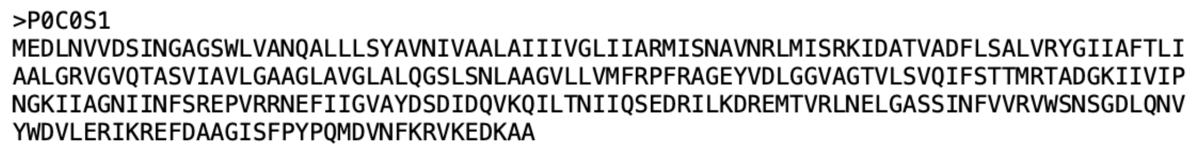


Figure 2: General Structure of an Alpha Amino Acid
The chemical structure of the amino acids showing side chain [Fla19].

Table 1: Amino Acid Letter Codes

| Amino Acids | Three Letter Code | Single Letter Code |
|---|---|---|
| Glycine | GLY | G |
| Alanine | ALA | A |
| Valine | VAL | V |
| Leucine | LEU | L |
| IsoLeucine | ILE | I |
| Threonine | THR | T |
| Serine | SER | S |
| Methionine | MET | M |
| Cystein | CYS | C |
| Proline | PRO | P |
| Phenylalanine | PHE | F |
| Tyrosine | TYR | Y |
| Tryptophane | TRP | W |
| Histidine | HIS | H |
| Lysine | LYS | K |
| Argenine | ARG | R |
| Aspartate | ASP | D |
| Glutamine | GLN | Q |
| Glutamate | GLU | E |
| Asparagine | ASN | N |



```
>P0C0S1
MEDLNVVDSINGAGSWLVANQALLLSYAVNIVAALAIIIVGLIIARMISNAVNRLMISRKIDATVADFLSALVRYGIIAFTLI
AALGRVGVQTASVIAVLGAAGLAVGLALQGSLSNLAAGVLLVMFRPFRAGEYVDLGGVAGTVLSVQIFSTTMRTADGKIIVIP
NGKIIAGNIINFSREPVRRNEFIIGVAYDSDIDQVKQILTNIIQSEDRILKDREMTVRLNELGASSINFVVRVWSNSGDLQNV
YWDVLERIKREFDAAGISFPYPQMDVNFKRVKEDKAA
```

Figure 3: A Protein Sequence in Fasta format

A sequence in FASTA format begins with one line of description, followed by lines of sequence data. The definition line is distinguished from the sequence data by a greater-than (>) symbol at the beginning. The word following the " > " symbol is the identifier of the sequence, and the rest of the line is the description which is optional.

and export metabolic products out. They also process a similar exchange for intracellular organelles.

There are three main types of integral membrane proteins; channels, transporters and ATP powered pumps. Channels transport water, ions, or hydrophilic small molecules by their concentration or electric potential gradients. Transporters are involved with the movement of ions, small molecules, and macromolecules. They allow certain substrates enter or leave the cells. ATP-powered pumps move ions or small molecules against a chemical concentration gradient, an electric potential, or both.

## 2.2 Protein Sequence Analysis

The transporters are classified into families based on the transporter classification (TC) system. The assignment into a TC family can provide an indication of the transport mechanism but not the substrate specificity of the protein, since proteins belonging to the same TC family transport different substrates and proteins belonging to different families can transport the same substrate. Classifying the transporter at a level of substrate class is very challenging, since it is dependent on a very small number of sites in the protein sequence, and those sites are not known beforehand. The structure and function information available for membrane proteins is very limited. Experimentally finding a function of a protein is a hard task. Hence, we make use of the available experimental data and the transmembrane transport protein sequences in computational tools to predict the transmembrane and their function. By classifying membrane proteins, we can understand their functionality.

### 2.2.1 Multiple Sequence alignment

Sequence alignment is a way of arranging the residues different sequences with respect to each other. It is used to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences. Sequences can be aligned either by performing local alignment or global alignment. Local alignment aligns the substring of the sequences, whereas global alignment aligns all of the residues of every sequences. The alignment of two sequences is called pairwise alignment and the alignment of three or more sequences is called multiple sequence alignment (MSA).

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

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

.

Given a set of $N$ sequences $S : S = \{S_1, S_2, ..., S_N\}$, $N \geqslant 2$, $S_i = S_{i1}, S_{i2}, ..., S_{il_i}$, for $i \leqslant N$, 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 described by inserting gaps into sequences $S_i$ in $S$ that the modified sequences $S'_i$ in set $S'$ conform to length $L$, where $L \geqslant max\{l_i | i = 1, ..., N\}$ or in other words, MSA is defined as a matrix $A = (a_{ij})$, $1 \leqslant i \leqslant l$, $max(l_i) \leqslant l \leqslant \sum_{i=1}^{N} l_i$. 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.

In order to measure the similarity between multiple sequences, a scoring mechanism is defined that is based on the multiple sequence alignment. All of the multiple sequence alignment methods use a metric called the objective function to measure the quality of their method. The most common objective function used is the sum-of-pairs function (SP).

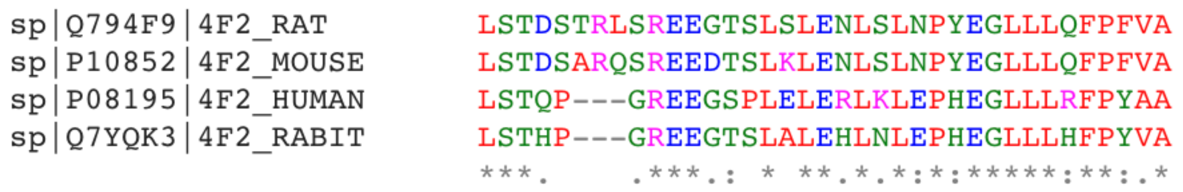
The SP function gets two important arguments: substitution matrix M and gap penalties (including open gap penalties and extension gap penalties).

The SP function calculates the sum of the residue scores and the penalty scores. The higher the result, the better alignment. So the MSAs should aim to maximise matches while permitting minimum gaps.

$$SP(S_1, S_2, ..., S_N) = \sum_{h=1}^{L} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} cost(S_{ih}, S_{jh})$$

where

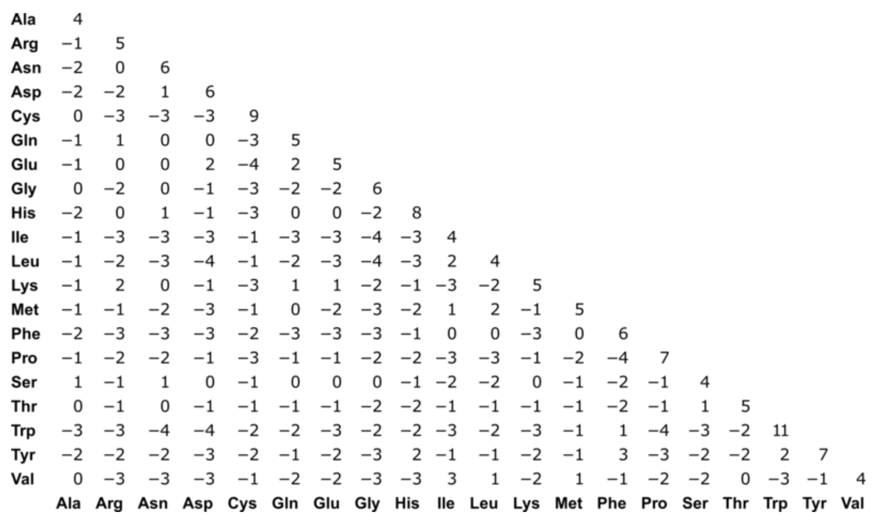
$$cost(S_{ih}, S_{jh}) = \begin{cases} M(S_{ih}, S_{jh}) & \text{If both are the same residue (match);} \\ M(S_{ih}, S_{jh}) & \text{If both are residues, but different (non-match);} \\ 0 & \text{If either is a gap.} \end{cases}$$



```
sp|Q794F9|4F2_RAT           LSTDSTRLSREEGTSLSLENLSLNPYEGLLLQFPFVA
sp|P10852|4F2_MOUSE         LSTDSARQSREEDTSLKLENLSLNPYEGLLLQFPFVA
sp|P08195|4F2_HUMAN         LSTQP---GREEGSPLELERLKLEPHEGLLLRFPYAA
sp|Q7YQK3|4F2_RABIT         LSTHP---GREEGTSLALEHLNLEPHEGLLLHFPYVA
                            ***.    .***.: * **.*.*:*:*****:**:.*
```

Figure 4: Local alignment of multiple sequences with BLAST
An * (asterisk) indicates positions which have a single, fully conserved residue. A . (period) indicates conservation between groups of weakly similar properties. A : (colon) indicates conservation between groups of strongly similar properties

The substitution matrix is used to records the similarity of two residues in protein sequence alignment, therefore it is used to calculate the cost function. Point accepted mutation (PAM) and Blocks substitution matrix (BLOSUM) are amongst the most commonly used substitution matrices [HH92]. The most commonly used are the PAM250 matrix and the BLOSUM-62 matrix. The BLOSUM-62 similarity score matrix is equivalent to a 62% residue match between the two sequences. Since multiple sequences in the alignment are



| | Ala | Arg | Asn | Asp | Cys | Gln | Glu | Gly | His | Ile | Leu | Lys | Met | Phe | Pro | Ser | Thr | Trp | Tyr | Val |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ala | 4 | | | | | | | | | | | | | | | | | | | |
| Arg | −1 | 5 | | | | | | | | | | | | | | | | | | |
| Asn | −2 | 0 | 6 | | | | | | | | | | | | | | | | | |
| Asp | −2 | −2 | 1 | 6 | | | | | | | | | | | | | | | | |
| Cys | 0 | −3 | −3 | −3 | 9 | | | | | | | | | | | | | | | |
| Gln | −1 | 1 | 0 | 0 | −3 | 5 | | | | | | | | | | | | | | |
| Glu | −1 | 0 | 0 | 2 | −4 | 2 | 5 | | | | | | | | | | | | | |
| Gly | 0 | −2 | 0 | −1 | −3 | −2 | −2 | 6 | | | | | | | | | | | | |
| His | −2 | 0 | 1 | −1 | −3 | 0 | 0 | −2 | 8 | | | | | | | | | | | |
| Ile | −1 | −3 | −3 | −3 | −1 | −3 | −3 | −4 | −3 | 4 | | | | | | | | | | |
| Leu | −1 | −2 | −3 | −4 | −1 | −2 | −3 | −4 | −3 | 2 | 4 | | | | | | | | | |
| Lys | −1 | 2 | 0 | −1 | −3 | 1 | 1 | −2 | −1 | −3 | −2 | 5 | | | | | | | | |
| Met | −1 | −1 | −2 | −3 | −1 | 0 | −2 | −3 | −2 | 1 | 2 | −1 | 5 | | | | | | | |
| Phe | −2 | −3 | −3 | −3 | −2 | −3 | −3 | −3 | −1 | 0 | 0 | −3 | 0 | 6 | | | | | | |
| Pro | −1 | −2 | −2 | −1 | −3 | −1 | −1 | −2 | −2 | −3 | −3 | −1 | −2 | −4 | 7 | | | | | |
| Ser | 1 | −1 | 1 | 0 | −1 | 0 | 0 | 0 | −1 | −2 | −2 | 0 | −1 | −2 | −1 | 4 | | | | |
| Thr | 0 | −1 | 0 | −1 | −1 | −1 | −1 | −2 | −2 | −1 | −1 | −1 | −1 | −2 | −1 | 1 | 5 | | | |
| Trp | −3 | −3 | −4 | −4 | −2 | −2 | −3 | −2 | −2 | −3 | −2 | −3 | −1 | 1 | −4 | −3 | −2 | 11 | | |
| Tyr | −2 | −2 | −2 | −3 | −2 | −1 | −2 | −3 | 2 | −1 | −1 | −2 | −1 | 3 | −3 | −2 | −2 | 2 | 7 | |
| Val | 0 | −3 | −3 | −3 | −1 | −2 | −2 | −3 | −3 | 3 | 1 | −2 | 1 | −1 | −2 | −2 | 0 | −3 | −1 | 4 |

Figure 5: BLOSUM62 [Com20a]

not identical, a **gap** is introduced to define insertion and deletion. To avoid aligning non homologous sequences, the number of gaps is limited to some extent. This number is retrieved by the scoring strategy the algorithm use. The matched residues get a positive score, while the gaps get a negative score or a penalty. There are two types of **gap penalty**: the **gap open penalty** and the **gap extend penalty**. The gap open penalty refers to the first gap inserted in a sequence while the sequence is being aligned and it normally has higher penalty than the gap extend penalty. The gap extend penalty refers to the insertion of the consecutive gap after the first gap is inserted.

Although there is no single best technique to find the best alignment, finding the alignment with biological correctness is not an easy task. When the number of sequences and length of sequences increase, finding the best exact alignment is too complex and computationally expensive. Therefore, MSA methods use heuristic technics to find the best alignment. The followings are the main types of heuristic algorithms used in MSA methods:

- Progressive algorithm
- Iterative algorithm
- Consistency-based algorithm

The idea of the **progressive algorithm** is to take an initial, approximate, phylogenetic tree between the sequences and to start with the two closest sequences and then gradually build up the alignment, following the order in the tree. Although it is a simple and fast algorithm, it is a greedy approach that its errors in the first alignments cannot be corrected later when the rest of the sequences are added which results in a "local minimization" problem. The representative progressive algorithms include ClustalW and Clustal Omega.

The concept of **Iterative alignment algorithm** is to produce progressive alignment and then fix the errors in the initial alignment iteratively untill the alignment results are no longer improved. The difference between this method and the progressive algorithm is that the whole pairwise alignments produced in the beginning of it can be revisited and modified to get a better score. Iterative alignment algorithm is not as fast and efficient as the progressive algorithm and it does not provide guarantees for obtaining optimized results. However, in this method, the objective function and the optimization process are theorotically separated and it is robust and insensitive to the number of sequences. MAFFT and MUSCLE are two examples of that implement this algorithm.

**Consistency-based methods** main idea 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 with residue pairs from all of the other alignments is examined and weighted so it reflects the degree to which those residues align consistently with other residues. Based on benchmarking studies [Pev09b], the final multiple sequence alignments generated by this method are more accurate than the ones achieved by progressive alignments. T-Coffee is considered a representative of this algorithm.

### 2.2.2 Specificity Determining Sites

Gene duplication cause the accumulation of amino acid changes which can result in functional divergence [Gu01]. Proteins that share common evolutionary origins in a species with different function than each other are called paralogs. However, most amino acid changes represent neutral evolution and not related to functional divergence [Kim91].

Generally, the more important a function of amino acid site is, the less possible it is 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 [Kim91]. A specificity determining site is the site in the amino acid sequences that any changes in its amino acids cause changes in the protein function. The SDS have special conservation or evolutionary role within a protein family [CC15], comparing to the neutral, non-function-related mutations in the majority of amino acid sites.

There are three types of functional divergence. Because of the pattern of evolution in them, several **evolutionary rate-base approaches** to SDS prediction have been defined. **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 [Kim91]. The site of type I functional divergence is different between two subfamilies. 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 different amino acid types for different protein subfamilies. **Type III** or **type MC** (marginally conserved) is nvolved in subfamily specificity determination where no apparent conservation of amino acids is observed within any of the subfamilies [CBP07].

Chakrabarti et al. [CBP07] indicate that almost half of the SDS belong to the type MC. Therefore, finding SDS using only conservation/evolution information from a strong background signals is extremely difficult. During the past decades, different algorithms has been developed, including the comination of at least some the signals from MSA, phylogenetic tree, amino acid physico-chemical properties and the protein's 3D structure [CC15]. Other computational algorithms are based on three approaches: evolutionary rate-based which was mentioned earlier, entropy-based, and amino acid physicochemical properties-based.

Several computational approaches have been developed to identify functional sites that could be involved in specific function related to a subset of proteins within a family over the last couple of decades. They relied on the conservation pattern of amino acids in a protein sequence alignment, together with structural constraints as indicators of likely functional importance and are mainly use the concept of principal component analysis, systematic use of physico-chemical properties of amino acids and the widely popular evolutionary trace (ET) method for SDS prediction [CC15]. However, these methods were more biased toward the type II SDS.

Some of the other algorithms use the mutual information (MI) to differentiate SDS alignment columns from non-SDS columns. These algorithms are among the **Entropy-based approaches**. However, the direct use of relative entropy to identify SDS were introduced later. Entropy-based approches calculate the relative entropy in one position between two subfamilies by computing the disturbution of each amino acid in each colomn. A cumulative relative entropy (CRE) is the sum of relative entropy in one position between two subfamilies. The cumulative relative entropy is normalized to achieve Z-score. The larger the Z-score value, the more the amino acid distributions in both subfamilies differ from one another.

**Physico-chemical properties-based approaches** can capture amino acid properties in different ways. One of them is discussed in Section 2.7.2

## 2.3 Profile Hidden Markov Model

The hidden Markov model (HMM) is a kind of Markov chain. It describes a probability distribution over a potentially infinite number of sequences. Since these probabilities must sum to one, there is a constraint on the score that the HMM assigns to sequences. Also, if the probability of one sequence increases/decreases, the probability of one or more of the sequences should decrease/increase. The states of the hidden markov model cannot be directly observed but they can be observed with the observation vector sequences that are described in every state by the probability distribution. The states sequences constitute a first-order Markov chain; meaning that the probability of one state depends only on its previous state. An HMM is a type of a non-deterministic finite state machine with transiting to another state and emitting a symbol under a probabilistic model.

The hidden markov model is a 5-tuple model: (S,O,T,E,I) where :

1. S: A finite set of states S=$\{q_1, q_2, ..., q_N\}$

2. O: A finite set of observed values O=$\{O_1, O_2, ..., O_M\}$

3. T: A probability of transition T=$\{t_{ij}\}$

4. E: An emission probability

$$E = e_{ik}, e_{ik} = P(O_t = O_k | S_t = q_i) \tag{1}$$

   which means the probability of seeing $O_k$ when in state $q_i$.

5. I: An initial state distribution I=$\{i_j\}$, $i_j$ = P($S_1 = q_i$).

A profile hidden markov model has a start state and an end state which do not emit any symbols. In the profile hidden markov model, each node is one of the following:

- Matching state (m): one character in the column.

- Insertion state (i): additional characters can be emitted between the columns.

- Deletion state (d): no characters are emitted in the column.

The main steps to create a profile HMM:

1. Determine the matching status (main status). In the multiple sequence alignment of one protein family, the gaps tend to be line up, so the model considers the ungapped regions. The probability of a new sequence $x$ according to this model is $P(x|M) = \prod_{i=1}^{l} e_i(x_i)$. To avoid underflow, it tried to evaluate $S = \sum_{i=1}^{l} log\frac{e_i(x_i)}{q_{x_i}}$, 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).

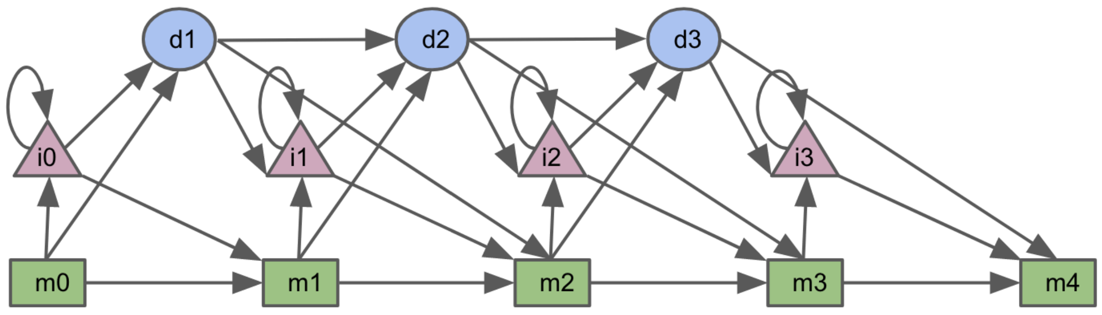


Figure 6: Different states of HMM and their transitions.
The green states represent the Matching residues. The pink states are the Insertation states and the blue states are Deletion states.

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.

### 2.3.1 Viterbi Algorithm

The **Viterbi algorithm** answers the question of given a known string, what is the sequence most likely to produce it, which is the **decoding problem**. The amount of time for calculating the possibility of all the possible state sequences will be exponential. The Viterbi algorithm tries to solve it with dynamic programming.

Definition $\phi_{t+1}(S_k) = \underset{l=1,2,...,N}{arg\,max}[\delta_t(l)T(l,k)], t = 1,2,...,n-1, k = 1,2,...,N$. 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 $s_n^* = \underset{l=1,2,...,N}{arg\max}[\delta_t(l)T(l,k)], s_t^* = \phi_{t+1}(S_{k+1}^*), t = n-1,...,1$, 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 $V_{i+1,j} = \max_{0\leq k\leq j-1}(V_{i,k}P(k,j)P(q_{i+1}|j))$

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

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

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

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 $x_{1,\ldots,i}$ 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 $D_j$.

## 2.4 Bioinformatic Protein Database

The Universal Protein Resource (UniProt) is an accessible database of protein sequence and functional information and is mainly supported by the National Institutes of Health (NIH) The UniProt databases are the UniProt Knowledgebase (UniProtKB), the UniProt Reference Clusters (UniRef), and the UniProt Archive (UniParc).

The `UniProtKB` is the primary worldwide database of protein sequences with accurate, consistent and rich annotation. Each protein record contains a list of keywords that summarizes the content of a `UniProtKB` entry and facilitates the search for proteins of interest. The keywords are controlled vocabulary with a hierarchical structure that are added during the manual annotation process. The `UniProtKB` consists of two parts:

- `Swiss-Prot`: contains manually-annotated, nonredundant records with information extracted from literature and curator-evaluated computational analysis. The annotation includes the protein and gene name, keyword assignment, function, subcellular location, peer-reviewed references, secondary structure elements, cross-references to other biological databases and information about their function

- `TrEMBL`: This section contains unrevised and automatically annotated protein sequences that await full manual annotation.

## 2.5 Bioinformatics Tools for Alignment

### 2.5.1 BLAST

Basic Local Alignment Search Tool (Blast)[AGM$^+$90] is a sequence alignment algorithm based on the idea that similar proteins must have short matches. It is faster than most of the sequence alignment algorithms that align a query sequence against all sequences in a sequence database to find similar sequences or matches because sequence databases can contain millions of sequences making optimal alignments computationally expensive. Blast generates all possible short words or substrings of the query sequence. The default length of a word for protein sequences is 3 and for nucleic acid sequences is 11. The algorithm scans a sequence database for sequences that match the words with some threshold. Such matches are called seeds. The original Blast then extends the seeds to the right and left using ungapped alignments[AMS$^+$97]. The algorithm terminates when the score of the extended alignment falls below some threshold S. Blast reports the extended alignments or hits that have a score at least S with their statistical significance. Such hits are called High Scoring Pairs (HSPs).

Blast uses a substitution matrix to compute the scores of each HSP. Statistical analysis of Blast alignment scores have been performed in the literature. The statistical significance of a Blast score S is given by the expected number, E-value [PF01], of an HSP with a score equivalent to or better than S. The probability of finding exactly n HSPs with score >=S is called P-value which is calculated as follows:

$$p = e^{-E}\frac{E^n}{n!} \tag{2}$$

Where E is the E-value of S. The BLAST programs report E-value rather than P-values because it is easier to understand the difference between. However, when E $<$ 0.01, P-value and E-value are nearly identical. The lower the E-value, the more significant the score and the alignment are. For a pair of query and subject sequences, Blast reports all HSPs and their associated measurements. The measurements of interest for the purpose of this document are query coverage, subject coverage, percent identity, E-value, and score. Query coverage is the ratio of the length of the HSP in the query sequence to the full length of the query sequence. Subject coverage is the ratio of the length of the hit in the subject sequence to the full length of the subject sequence. Percent identity is the percentage of identical amino-acids at the same positions in the alignment with respect to the alignment length. Score is the bit score, which is the raw score calculated from the substitution matrix normalized to parameters including the database size.

The program used the following parameters: Maximum target sequences: 120; E-value: 0.001; Word Size: 3 as Default; 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 [THG94] is one of the most popular MSA algorithms that uses a progressive alignment algorithm. It starts with performing a pairwise alignment of all the sequences in the alignment in a matrix that shows the similarity of each pair of sequences. Then it construct a phylogenetic tree called a guide tree using the neighbor-joining method [SN87] and the distance score matrix which can be interpreted by the similiarity score matrix. Afterwards it starts with the two most similar sequences. Finally, it gradually adds new sequences until all sequences are added.

In progressive methods, Different order of addition will produce different results. Therefore, finding the right alignment order is the key issue. Although progressive methods are very efficient, they can not correct an error that's been made at the early stages of the alignment which can increase the likelihood of misalignment due to incorrect conservation signals [DOS13] [Pev09a].

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 [SWD+11] is another tool in the Clustal family that uses progressive alignment algorithm. It works faster and more accurate than ClustalW. It also provides features that can be used to avoid recomputing an entire alignment every time that new sequences become available. Clustal Omega constructs the guide tree using a modified version of mBed [BSS+10] which is embedding the sequences in a vector space of $n$ dimension where each element is the distance to the corresponding sequence. The pairwise alignment are computed using the HHalign package [Sö05] which aligns the sequences with two profile hidden Markov models [Edd98]. For an alignment of N sequences, mBed algorithm hast the complexity of $\mathcal{O}(NlogN)$.

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

### 2.5.4 MAFFT

The multiple alignment using fast Fourier transform (MAFFT) algorithm [KMKM02] is an iterative method that uses two-cycle heuristics. The frequency of amino acid substitutions strongly depends on the difference of physico-chemical properties. Initially each amino acid is converted into a vector, which includes two values: volume and polarity. The algorithm calculates the correlation between two amino acid sequences, the result is transformed to Fourier signal information, which reduces the cpu time from $\mathcal{O}(N^2)$. to $\mathcal{O}(NlogN)$. If two sequences compared have homologous regions, their correlation will has some peaks corresponding to these region. MAFFT applies standard dynamic programming to obtain an optimal path, which corresponds to the optimal arrangement of similar segments. In the program package MAFFT, there are several different modes including progressive methods FFT-NS-1 and FFT-NS-2, and an iterative refinement method FFT-NS-i and NW-NS-i. FFT-NS-1 and FFT-NS-2 use a guide tree like ClustalW. FFT-NS-i divides the alignment into two groups and realigns until there is no improvement in score. In the NW-NS-i algorithm the FFT approximation is disabled.

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

### 2.5.5 MUSCLE

Multiple Sequence Comparison by Log-Expectation(MUSCLE) [Edg04] is a combination of the progressive method and iterative method. MUSCLE has three stages. At the completion of each stage, a multiple alignment is available and the algorithm can be terminated. The first stage starts by computing the similarity of each pair of sequences using the k-mer counting or by constructing a global alignment of the pair and determining the fractional identity. A tree is constructed from the distance matrix using UPGMA or neighbor-joining, and a root is identified. By progressive alignment, MUSCLE makes a preliminary alignment, MSA1. The second stage, which is the iterative method, the algorithm tries to improve the tree and builds a new progressive alignment according to this tree. The similarity of each pair of sequences is computed using fractional identity computed from their mutual alignment in the current multiple alignment. This time the tree is constructed by computing a Kimura distance matrix and applying a clustering method to this matrix. Both trees are then compared to identify the set of internal nodes for which the branching order has changed. If the these step of the second stage are executed more than once and the number of changed nodes has not decreased, the process of improving the tree is considered to have converged and iteration terminates. Then it builds another progressive alignment. In the third stage, an edge is deleted from the guide tree to divide The sequences into two groups. The profile of the multiple sequence alignment in each subtree is computed. The two profiles are re-aligned to each other using profile-profile alignment. If the score of the new alignment is improved, the new alignment is kept, otherwise it is discarded.

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

### 2.5.6 AQUA

Automated Quality Improvement For Multiple Sequence Alignments or simply AQUA combines existing tools in three steps: the first step is called Computation of an initial MSA. It aligns the sequences with MAFFT and MUSCLE. The second step or Refinement of the MSA is introduced to detect and correct the errors made in the previous step. The program uses RASCAL [TTP03]. The last step or Evaluation and selection of the best MSA uses the NORMD [TPTP01] program to estimate the quality of each MSA produced in the previous steps. The NorMD score gives information about the general quality of the alignment but can also be used to compare different versions of the same MSA. Thus, it selects the MSA with the highest NorMD value.

AQUA can be downloaded from: `http://www.bork.embl.de/Docu/AQUA`.

### 2.5.7 T-Coffee

T-Coffee (Tree-based Consistency Objective Function for alignment Evaluation) is also a greedy progressive method but it has two main features that provides it with better information in the early stages. First, it allows that the multiple alignments be generated by different ways using heterogeneous data sources. These data are provided to T-Coffee by a library of pairwise alignments. The second one is the optimization method it uses. The optimization method tries to find the multiple alignments that best fits the pairwise alignment which is provided in the input library. It uses a progressive strategy, that is, using the information in the library to carry out progressive alignment in a manner that it considers the alignments between all the pairs. It has the advantage of being fast and accurate. T-Coffee is a progressive alignment with an ability to consider information from all of the sequences during each alignment step, not just those being aligned at that stage.

T-Coffee algorithm has five steps. The first one is generating a primary library of alignments. In this step, T-Coffee uses two alignment sources for each pair of sequences, one is local (Lalign [HM91]) and the other one is global using ClustalW. 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 pairwise 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 pairwise 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)). Lastly, the progressive alignment strategy uses the the extension library as substitution matrix to align the sequences according to the phylogenetic tree [SN87].

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

### 2.5.8 TM-Coffee

Transmembran 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. Most of the multiple alignment techniques use scoring schemes tailored for soluble proteins and are therefore, may not produce the optimal alignment for transmembrane transport proteins [PFH08]. TM-Coffee [FTC$^+$16] is designed for this situation by using homology extension. In homology extension methods, database searches are used to replace each sequence with the profile of closely related homologs. Although this method gives much more accurate alignment, performing an alignment takes several orders of magnitude longer than the standalone applications [EB06].

The TM-Coffee algorithm is described as follows. First, run BLAST against a specified database for each sequence that is going to be aligned to find homologous sequences with

50% to 90% identity and a coverage of more than 70% are retaines. Then the BLAST output is turned 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 [FTC+16]. 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 Secator for Protein Subfamilies

With the increase in protein database sizes, finding the the number of sensible subsets of a large protein family for in-depth structural, functional, and evolutionary analyses is an important task. Secator [WPTP01] is a program that implements the principle of an ascending hierarchical method using a distance matrix based on a multiple alignment of protein sequences. The algorithm follows. Initially, each sequence is considered a family of protein and the dissimilarity between each pair is calculated based on the distance matrix given as an input and weight of the sequences. Secondly, while the number of families is greater than two do: it combines the two families with the smallest dissimilarity value and compute the dissmilarity and weights of the new formed family and the rest of the families. Once the number of families are two, Secator assign the dissimilarity of the two remaining families to a virtual node. The nodes are clustered into two groups, the group with high dissimilarity values and the group with low dissimilarity values. This clustering is done by computing the partition into two groups which has the maximum interclass inertia on a subset of all possible partitions. This partitioning produces a threshhold for the low dissimilarity value. The families with the dissimilarity value below the threshhold are partitioned and the new interclass inertia is calculated. The algorithm continues this procedurte all of the dissimilarity values are higher than the threshhold. Finally, all of the nodes of a cluster with dissimilarity values above the threshhold are put into a family.

Secator can be downloaded from: `http://math.univ-lille1.fr/~wicker/softwares.html`.

## 2.7 Bioinformatics Tools for Specificity Determining Sites

### 2.7.1 GroupSim

GroupSim [CS08] 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 [CC15]. 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, the average similarity between each pair of amino acids in a subfamily is calculated according to a similarity matrix for each subfamily in the alignment. Second, in order to differentiate between specificity groups, 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. Finally, GroupSim calculates the average of each subfamily similarity average. 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/`.

### 2.7.2 SPEER-SERVER

SPEER is an ensemble approach for specificity prediction by analyzing quantitative measures of the conservation patterns of protein sites based on their physico-chemical properties and the heterogeneity of evolutionary changes between and within the protein subfamilies [CBP07].

The first component of SPEER, `PC property distance` calculates the weighted Euclidean distance between the vectors of physico-chemical properties of any two positions in the MSA. The average variability in a given subfamily column is calculated by summing the Euclidian distances for all residue pairs within the subfamily and normalizing by the average Euclidian distance of all residue pairs in the column of the overall family. 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 component is the `Evolutinary rate` (ER) of the site as computed by the maximum-likelihood method implemented in the rate4site [PBM$^+$02] program. A low average ER value indicates a strongly conserved site in subfamilies. The last component, `relative entropy`, is used to quantitatively distinguish the amino acid-type distributions of two protein subfamilies by calculating 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 as default 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`.

### 2.7.3 TCS

The TCS (transitive consistency score) [CDTN14] is an alignment evaluation score that makes it possible to identify in an MSA the most correct positions. It uses a consistency transformation to assign a reliability index to every pair of aligned residues, to each individual residue in the alignment, to each column, and to the overall alignment. This scoring scheme has been shown to be highly informative with respect to structural predictions based on benchmarking databases. The reliability index ranges from 0 to 9, where 0 is extremely uncertain and 9 is extremely reliable. Columns with a reliability index less than 4 are removed.

TCS can be download from: `http://www.tcoffee.org/Projects/tcs/`.

### 2.7.4 Xdet

The idea behind Xdet [PRV06] is that, in the specificity sites, a sharp amino acid change between two proteins would be related with a high functional difference between these proteins, and the other way around. Xdet implements two methods for detecting residues responsible for functional specificity in multiple sequence alignments. The first method is the mutational behaviour (MB) method, and the second one is an upgraded version of the MB-method, which uses an external arbitrary functional classification instead of relying on the one implicit in the alignment. 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, Xdet construct a matrix to calculate the amino acid changes for all pairs of proteins is constructed based on a substitution matrix. In this matrix, a given entry represents the similarity between the residues of two proteins at that position. The second method, starts by constructing an equivalent matrix from an external explicit functional classification where each entry represents the functional similarity between the corresponding proteins. These two matrices are compared with a Spearman rank-order correlation coeficient

$$r_k = \frac{\sum_{pq}({A_{ijk}}' - \overline{A}')({F_{ij}}' - \overline{F}')}{\sqrt{\sum_{ij}\left({A_{ijk}}' - \overline{A}'\right)^2}\sqrt{\sum_{ij}\left({F_{ij}}' - \overline{F}'\right)^2}} \tag{3}$$

where $r_k$ is the score for position $k$, $A_{ijk}$ is the similarity between the amino acids of proteins $i$ and $j$ at position $k$, and $F_{ij}$ is the functional similarity between proteins $i$ and $j$. 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 [PRV06].

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

## 2.8 HMMER for Profile Hidden Markov Model

HMMER [FCE11] is a free and commonly used software package used for searching sequence databases for sequence homologs, and for making sequence alignments. It implements methods using probabilistic models called profile hidden Markov models. There are patterns of site-specific evolutionary conservation when aligning multiple sequence of a homologous family of protein domains. These conserved sites correspond to functional sites. The non-conserved sites are assumed to be non-specific feature sites. Therefore, There are different probability distribution sites 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 [KBM+94].

HMMER uses `hmmbuild` to build a profile HMM using a multiple sequence alignemnt or single sequence as input. A profile is a position-specific scoring model that describes which symbols are likely to be observed and how frequently insertions/deletions occur at each position (column) of a multiple sequence alignment. The hidden state is one of the insert state, delete state and match state, while the observation layer consists of sequence letters. 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. Afterwards, the software uses `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`.

# Chapter 3

# Classifier Construction

Existing tools that predict the substrate that is transported lag behind tools for annotation of other kinds of proteins such as enzymes. Most tools predict the type of substrates [SCH10, COLG11, SH12, BH13, MCZ14], chosen from a small subset of substrate types, or predict the family or subfamily [GY08, LBUZ09, OCG10, BH13] for the protein within the Transporter Classification (TC) [SJRT$^{+}$16]. None attempt to predict the specific substrate. The state-of-the-art tool for *de novo* prediction of substrate class was TrSSP [MCZ14], as benchmarked on their dataset with seven substrate classes, which has become a standard benchmark in the literature.

## 3.1 Materials

### 3.1.1 Datasets

The first dataset of TrSSP is based on the Swiss-Prot database (release 2013-03). It is a standard benchmark in the literature. It is available at http://bioinfo.noble.org/TrSSP. The dataset contains 900 sequences of seven substrate classes: *amino acid, anion, cation, electron, protein/mRNA, sugar*, and *other*, divided into the training set of size 780 and the test set of size 120 (see Table 2).

An updated second dataset was collected from Swiss-Prot database of June 2018 and carefully assigned to eleven substrate classes [AB19]. The dataset has 1524 sequences which

Table 2: Dataset One

| Substrate Class | Training | Test | Total |
|---|---|---|---|
| Amino Acid | 70 | 15 | 85 |
| Anion | 60 | 12 | 72 |
| Cation | 260 | 36 | 296 |
| Electron | 60 | 10 | 70 |
| Protein/mRNA | 70 | 15 | 85 |
| Sugar | 60 | 12 | 72 |
| Other | 200 | 20 | 220 |
| Total | 780 | 120 | 900 |

The table shows Dataset One [MCZ14] with its seven classes and the number of sequences in the training set and test set for each class.

are randomly divided into a training set of size 1376 and a test set of size 148 as is shown in Table 3. It is available at https://tootsuite.encs.concordia.ca/datasets.

Table 3: Dataset Two

| Substrate Class | Training | Test | Total |
|---|---|---|---|
| Nonselective | 24 | 2 | 26 |
| Water | 24 | 2 | 26 |
| Inorganic Cations | 541 | 60 | 601 |
| Inorganic anions | 92 | 10 | 102 |
| Organic anions | 97 | 10 | 107 |
| Organo-oxygens | 157 | 17 | 174 |
| Amino acids and derivatives | 142 | 15 | 157 |
| Other organonitrogens | 144 | 16 | 160 |
| Nucleotides | 22 | 2 | 24 |
| Organo heterocyclics | 34 | 3 | 37 |
| Miscellaneous | 99 | 11 | 110 |
| Total | 1376 | 148 | 1524 |

The table shows Dataset Two [AB19] with its eleven classes and the number of sequences in the training set and test set for each class.

### 3.1.2 Database

The database used for searching for similar sequences using BLAST is *Swiss-Prot* database. The *Swiss-Prot* database is also used by TM-Coffee to guide the construction of alignments. Hence, to maintain the independence between training and testing, we remove all the test sequences in the datasets from the databases used by BLAST and TM-Coffee.

## 3.2 Methods

For the training stage, we build a pipeline which combines different stages of data processing and a stage for building the model. The pipeline contains similarity search, multiple sequence alignment, specificity determining site prediction and construction of a profile Hidden Markov Model. Furthermore, the pipeline contains steps for model evaluation. Below are the steps of the training pipeline in order.

### 3.2.1 Blast

After placing each sequence in a separate file, we run BLAST [AGM+90] on each sequence in the training dataset to get similar sequences. For the Blast local database we use the Swiss-Prot by removing all copies of sequences in the test set. For each sequence in the training set, a BLAST search is performed to retrieve a maximum of 120 similar sequences using the following commandline:

```
BLAST blastp -query blast_query_path -db cleaned_swiss_prot_path -evalue
0.001 -outfmt 5 -out des_blast_file -max_target_seqs 120
```

**Algorithm 1** BLAST

```
Input: the training dataset as .fasta file of protein sequences;
Output: up to 121 similar sequences containing each sequence in the training dataset;
1: LocalDB ⇐ Search(Swiss-Prot, removing any copy of sequences in the train and test
   dataset)
2: for each sequence $s$ in training dataset do
3:     SimilarSeq[$s$] ⇐ blastp(DB=LocalDB, query=s, maxseq=120, e-value=0.001)
4:     SimilarSeq[$s$].append($s$)
5: end for
```

However, three sequences of Dataset One twenty three sequences of the Dataset Two did not have any hits with the database by an e-value of 0.001 or less.

### 3.2.2 Multiple Sequence Alignment

We adopted the MSA-SDS approach which finds specificity determining sites with the evolutionary information available from the MSA. This process is done by retrieving homologous sequences of each protein sequence in the dataset and then building an MSA for the set of similar proteins. The MSA methods used for this experiment are Clustalw [THG94], MAFFT [KMKM02], MUSCLE [Edg04], TM-Coffee [FTC+16], AQUA [MCT+09], ClustalOmega [SWD+11], and T-Coffee [NHH00] and the specific commandlines calls are as follows:

```
AQUA
    AQUA.tcl input_path output_path
ClustalOmega
    clustalo -infile=input_path -outfile=output_path  -output=fasta
ClastalW
    clustalw -infile=input_path -outfile=output_path -output=fasta
MAFFT
    mafft input_path > output_path
MUSCLE
    muscle -in input_path -out output_path
T-Coffee
    t_coffee -infile input_path -outfile output_path  -output fasta_aln}
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 Specificity Determining Sites

We filter an MSA to focus on important positions in the sequences. The SDS methods determine the important positions. We filter the alignment by changing all entries in the columns of non-SDS positions to '-', and other entries are unchanged. The SDS methods used are Speer Server [CML$^+$12], GroupSim [CS08], Xdet [PRV06], and TCS [CDTN14].

For Speer Server and GroupSim we run Secator [WPTP01] to separate the aligned sequences into subgroups based on a hierarchical clustering method. This is required as these SDS methods are based on the differences between subgroups. If Secator only returns one subgroup, then these SDS methods are bypassed for that training sequence.

**Algorithm 2** SpeerServer

```
Input: MSA in .fasta file;
Output: SDS of the input sequences in .fasta file;
 1: //finding the subfamilies
 2: subfamilies ⇐ secator(input=MSA, clustering_method=hierarchy)
 3: //prepare the input based on subfamilies in .fasta file, save the size of each subfamily
 4: size=[ ]
 5: for each subfamily g in subfamilies do
 6:     for each sequence s in subfamily g do
 7:         fMA.append(s)
 8:     end for
 9:     size.append(size(g))
10: end for
11: // Check the number of subfamilies
12: if length of size >1 then
13:     SDS ⇐ SPEER(input=fMA,sizes=size)
14: end if
15: for each position $p$ not in SDS do
16:     $fMA[r,p]$ ⇐ '-', for each row $r$ of alignment
17: end for
```

The command line of Secator is the following:

```
secator input_secator -dt=alignment -cm=hierar output
```

The commandline calls for the SDS methods are as follows:

GroupSim

```
python groupsim.py -n -m matrix_path -w 1 -o output_path
input_path sequences_in_group1 ... sequences_in_groupN
```

SpeerServer

```
SPEER -wRE 1 -wEDist 1 -wERate 1 -i input_path -o output_path
number_of_sequences_in_group1 ... number_of_sequences_in_groupN
```

**Algorithm 3** GroupSim

```
Input: MSA in .fasta file;
Output: SDS of the input sequences in .fasta file;
 1: //finding the subfamilies
 2: subfamilies ⇐ secator(input=MSA, clustering_method=hierarchy)
 3: //prepare the input based on subfamilies in .clustal file
 4: for each subfamily g in subfamilies do
 5:     for each sequence s in subfamily g do
 6:         fMA.append(s)
 7:     end for
 8: end for
 9: if length of size >1 then
10:     SDS ⇐ groupsim(input=fMA)
11: end if
12: for each position $p$ not in SDS do
13:     $fMA[r, p]$ ⇐ '-', for each row $r$ of alignment
14: end for
```

```
TCS

    t_coffee -infile MSA_output -outfile tcs_output -evaluate -output tcs_column_filter4.fasta

Xdet

    xdet input_path matrix_path > output_path -S 10
```

### 3.2.4 Hidden Markov Model

HMMER [FCE11] is a tool that constructs and utilises the profile Hidden Markov Models for the classification. We use *hmmbuild* to build a profile HMM from a multiple sequence alignment, or single sequence as input; we use *hmmpress* to compress the profiles for use in *hmmscan*; and we use *hmmscan* to scan (and score) each test sequence against each profile.

#### Train classifier

Training is done by building a HMM profile with *hmmbuild* from the filtered MSA. The commandline call is as follows:

```
hmmbuild output_path input_path
```

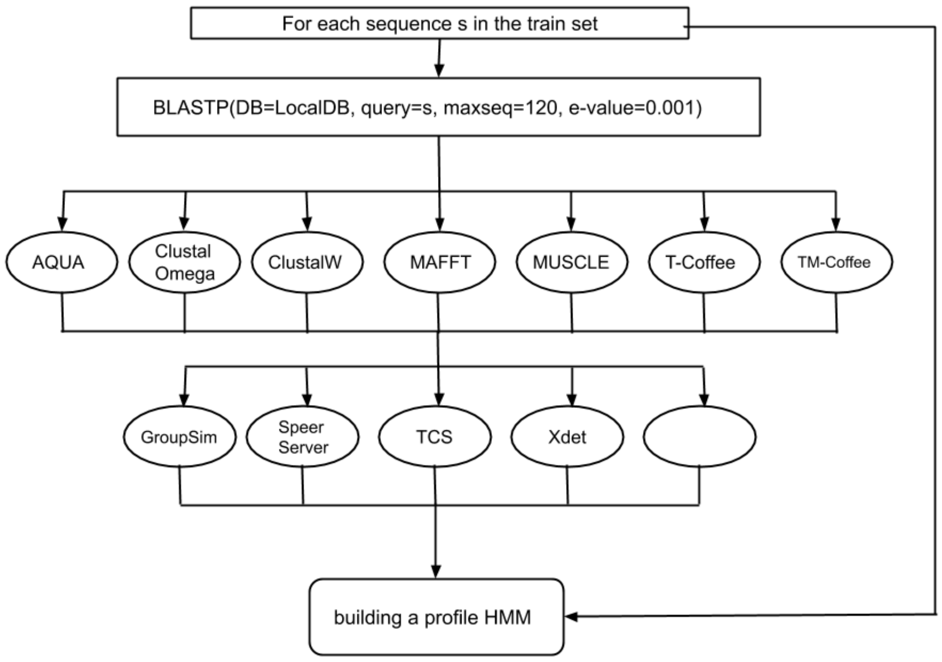


Figure 7: Training Steps
The figure shows the training steps of the pipeline. Note that each profile HMM can be built on the output of BLAST or the output of an MSA. These methods have their own processors; e.g. GroupSim contains running secator, preparing the GroupSim input, running GroupSim software and filtering out the unimportant columns of the MSA.

**Classification**

Classification is done by scoring each sequence of the test data against each HMM profile using *hmmscan*, and then selecting the class that corresponds to the profile with the best score. The commandline call is as follows:

```
hmmscan -o result_path -E 0.1 hmm_path query_path
```

### 3.2.5 Evaluation

Four statistical measures are used to measure the performance: Sensitivity, specificity, accuracy and Matthews correlation coefficient (MCC). For computing sensitivity, specificity, macro accuracy, macro MCC and overall MCC we disregard the unclassified sequences, but

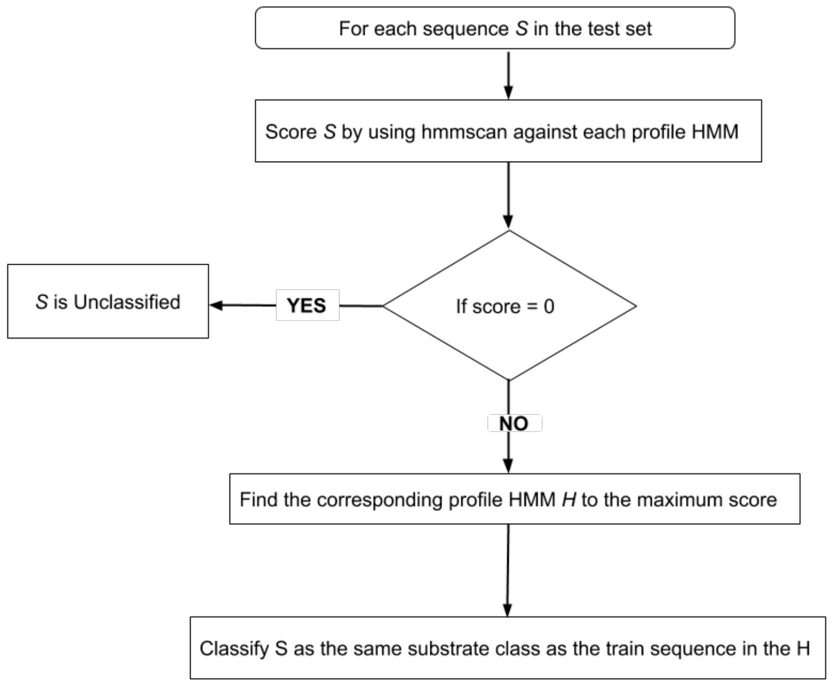

For each sequence S in the test set
Score S by using hmmscan against each profile HMM
If score = 0
YES
S is Unclassified
NO
Find the corresponding profile HMM H to the maximum score
Classify S as the same substrate class as the train sequence in the H

Figure 8: Classification Steps
The figure shows the classification steps of the pipeline. Note that each sequence in the test set is scored by running *hmmscan* against each profile HMM in the training set. *Unclassified* is defined as the test sequence that *hmmscan* outputs score 0 for all of the profile HMMs in the training set.

for calculating overall accuracy we take into the account of unclassified sequences.

**sensitivity** is the proportion of positive samples that are correctly identified:

$$Sensitivity = \frac{TP}{TP + FN} \tag{4}$$

In this measure we disregard the unclassified sequences. However, another option is to count the unclassified as $FN$ which would lower the current measurement. For calculating the *Sensitivity* of each class $k$, we define $TP$ or true positives as the sequences that were both belonged and predicted for class $k$ and $FN$ are sequences that are supposed to be in class $k$ but the predictor did not predict them as class $k$.

**specificity** is the proportion of negative samples that are correctly identified:

$$Specificity = \frac{TN}{TN + FP} \tag{5}$$

Where $TN$ is true negative which is the sequences that neither belong to class $k$ nor was predicted as class $k$ and $FP$ as false positives that are the sequences that did not belong to class $k$ but was predicted to be in class $k$.
**accuracy** is the proportion of correct classifications made amongst all the classifications:

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{6}$$

Which in our case can be extended to :

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP + Un} \tag{7}$$

Where $Un$ is the number of unclassified sequences and can be grouped into $FP$. Since the above equations are for binary classification, we use Equation 6 for calculating the accuracy of each class $k$, where $TP$ are the true prediction for class $k$, $TN$ are the sequences that neither belong to class $k$ nor was predicted as class $k$, $FP$ are sequences that did not belong to class $k$ but was predicted to be in class $k$ and $FN$ are sequences that are supposed to be in class $k$ but the predictor did not predict them as class $k$.
**Matthews correlation coefficient (MCC)** is a single measure taking into account true and false positives and negatives, and returns a value in the range from 1 to -1, where 1 indicates a perfect prediction, 0 represents prediction no better than random, and -1 implies total disagreement between the prediction and observation. Since MCC is less influenced by the imblanced tests, we use it as the best single assessment metric:

$$MCC = \frac{(TP \times TN - FP \times FN)}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \tag{8}$$

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. This formula however is for binary classification as well. As a result, for calculating the MCC of each class $k$ we used the above formula where $TP$ are the true prediction for class $k$, $TN$ are the sequences that neither belong to class $k$ nor was predicted as class $k$, $FP$ are sequences that did not belong to class $k$ but was predicted to be in class $k$ and $FN$ are sequences that are supposed to be in class $k$ but the predictor did not predict them as class $k$ . Another alternative to this calculation is to count the unclassified sequences as the false negative which in that case MCC score becomes smaller.

Since we are dealing with multiclass classification, a single aggregate measure that reflects the overall performance is needed. There are two methods to compute the overall performance, namely micro-averaging and macro-averaging. Macro-averaging computes a simple average performance of individual classes' performances. Whereas Micro-averaging computes an overall performance by globally counting the total true positives, false negatives and false positives. These methods can differ based on the class distribution. Macro-averaging is an unweighted average of each class, while micro-averaging gives equal weight to each individual classification decision. The overall accuracy of the tool is often calculated as the fraction of the correct predictions by the total number of predictions:

$$Accuracy_{Micro} = \sum_{k}^{K} \frac{TP_k}{N} \tag{9}$$

This measurement uses the number of Unclassifieds in $N$ since $N$ is the total number of sequences; $N{=}TP + FN + TN + FP + Un$. Also the macro-average accuracy is:

$$Accuracy_{Macro} = \frac{1}{K} \times \sum_{k}^{K} Accuracy_k \tag{10}$$

where $TP_k$ is the number of true positives in class $k$, $Accuracy_k$ is the accuracy of class k, $K$ is the number of different classes (7 and 11 for the respective datasets). It is worth mentioning that for calculating the accuracy of each class, $Accuracy_k$, we use Equation 6.

On the other hand, the overall MCC is calculated in terms of a $K \times K$ confusion matrix

C [Gor04]:

$$MCC_{overall} = \frac{\sum_k \sum_l \sum_m C_{kk}C_{lm} - C_{kl}C_{mk}}{\sqrt{\sum_k (\sum_l C_{kl})(\sum_{k'|k' \neq k} \sum_{l'} C_{k'l'})}\sqrt{\sum_k (\sum_l C_{lk})(\sum_{k'|k' \neq k} \sum_{l'} C_{l'k'})}} \quad (11)$$

Since this formula is based on a $K \times K$ confusion matrix, we disregard the unclassified sequences. The macro-average MCC is computed as:

$$MCC_{macro} = \frac{1}{K} \sum_{k=1}^{K} MCC_k \quad (12)$$

where $MCC_k$ is the MCC of class k, and $K$ is the number of different classes. Since the number of sequences in each class of the dataset is imbalanced, we used the overall accuracy as in Equation 9 and the overall MCC as in Equation 11 to evaluate and compare the different methods.

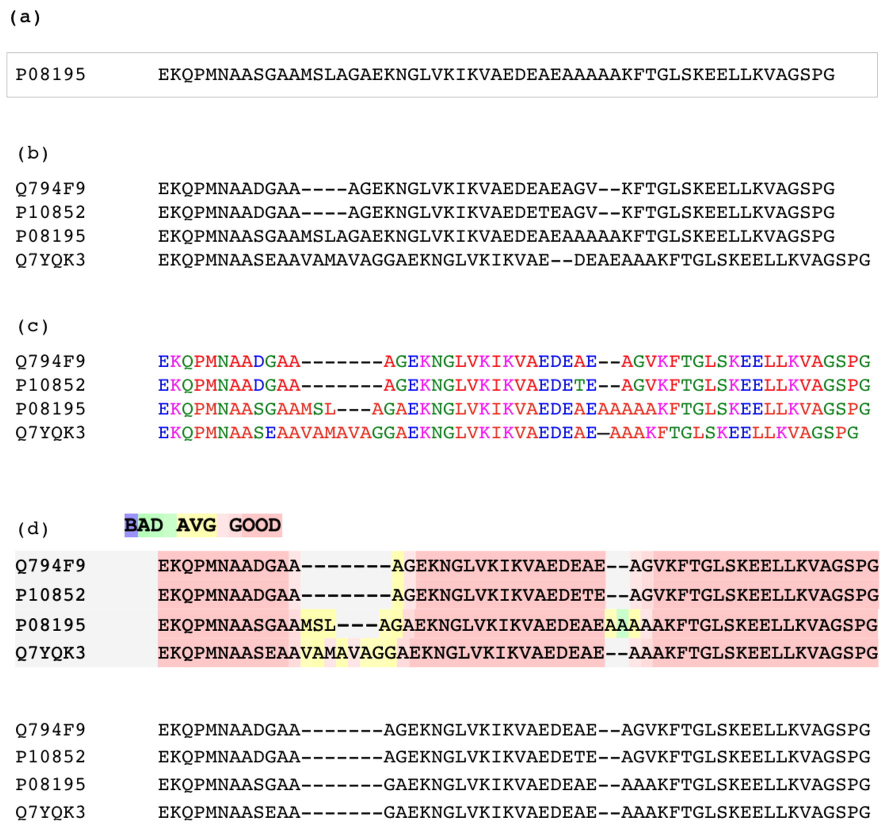
### 3.2.6 Experiment

In this experiment, we are trying to analyze the impact of different MSA and SDS tools. For both of the datasets, we consider each combination of MSA tool and SDS tool to determine the best combination. We also determine the impact of each MSA with or without using the SDS tools by including the baselines for each MSA tool. Furthermore, we include one primary baseline that does not use any MSA tool nor any SDS tool and simply builds a profile HMM from the (unaligned) set of similar sequences returned by the BLAST for each sequence. Figure 9 presents the steps of the pipeline before being fed to profile HMM.

## 3.3 Results

### 3.3.1 Comparative Performance

Table 4a shows the comparative performance of the 36 combinations of methods using Dataset One. They are sorted from low to high by overall MCC. The highest MCC was achieved by the primary baseline. Table 5a shows the comparative performance of the 36



```
(a)

P08195      EKQPMNAASGAAMSLAGAEKNGLVKIKVAEDEAEAAAAAKFTGLSKEELLKVAGSPG

(b)

Q794F9      EKQPMNAADGAA----AGEKNGLVKIKVAEDEAEAGV--KFTGLSKEELLKVAGSPG
P10852      EKQPMNAADGAA----AGEKNGLVKIKVAEDETEAGV--KFTGLSKEELLKVAGSPG
P08195      EKQPMNAASGAAMSLAGAEKNGLVKIKVAEDEAEAAAAAKFTGLSKEELLKVAGSPG
Q7YQK3      EKQPMNAASEAAVAMAVAGGAEKNGLVKIKVAE--DEAEAAAKFTGLSKEELLKVAGSPG

(c)

Q794F9      EKQPMNAADGAA-------AGEKNGLVKIKVAEDEAE--AGVKFTGLSKEELLKVAGSPG
P10852      EKQPMNAADGAA-------AGEKNGLVKIKVAEDETE--AGVKFTGLSKEELLKVAGSPG
P08195      EKQPMNAASGAAMSL---AGAEKNGLVKIKVAEDEAEAAAAAKFTGLSKEELLKVAGSPG
Q7YQK3      EKQPMNAASEAAVAMAVAGGAEKNGLVKIKVAEDEAE-AAAKFTGLSKEELLKVAGSPG

(d)    BAD AVG GOOD

Q794F9      EKQPMNAADGAA-------AGEKNGLVKIKVAEDEAE--AGVKFTGLSKEELLKVAGSPG
P10852      EKQPMNAADGAA-------AGEKNGLVKIKVAEDETE--AGVKFTGLSKEELLKVAGSPG
P08195      EKQPMNAASGAAMSL---AGAEKNGLVKIKVAEDEAEAAAAAKFTGLSKEELLKVAGSPG
Q7YQK3      EKQPMNAASEAAVAMAVAGGAEKNGLVKIKVAEDEAE--AAAKFTGLSKEELLKVAGSPG

Q794F9      EKQPMNAADGAA-------AGEKNGLVKIKVAEDEAE--AGVKFTGLSKEELLKVAGSPG
P10852      EKQPMNAADGAA-------AGEKNGLVKIKVAEDETE--AGVKFTGLSKEELLKVAGSPG
P08195      EKQPMNAASGAA-------GAEKNGLVKIKVAEDEAE--AAAKFTGLSKEELLKVAGSPG
Q7YQK3      EKQPMNAASEAA-------GAEKNGLVKIKVAEDEAE--AAAKFTGLSKEELLKVAGSPG
```

Figure 9: Example of steps of the Clustal Omega-TCS combination
Steps of the Clustal Omega-TCS combination. Part(a) shows a part of the original sequence. Part(b) shows the BLAST hits. Part(c) shows an MSA constructed by Clsuatl Omega. Part(d) demonstrates the steps of TCS, at first it evaluates each MSA position, then it filters out the positions that have less quality than the average.

combinations of methods using Dataset Two. They are sorted from low to high by overall MCC. The highest MCC was also achieved by the primary baseline as well.

### 3.3.2 Detailed Performance

We have defined TportHMM as the primary baseline where the profile HMM is built from a single sequence in the training set. This gives the best MCC in Table 4a and Table 5a. The detailed performance of the TportHMM on Dataset One and its confusion matrix are presented in Table 6 and Table 7 respectively, while Table 8 and Table 9 show its detailed performance and confusion matrix on Dataset Two.

Table 4a: Comparative Performance on Dataset One

| MSA | SDS | Micro Accuracy | Macro Accuracy | Overall MCC |
|---|---|---|---|---|
| T-Coffee | TCS | 0.033 | 0.846 | -0.087 |
| T-Coffee | SpeerServer | 0.233 | 0.840 | 0.212 |
| T-Coffee | GroupSim | 0.250 | 0.846 | 0.259 |
| ClustalOmega | SpeerServer | 0.308 | 0.849 | 0.326 |
| ClustalOmega | GroupSim | 0.342 | 0.851 | 0.331 |
| ClustalW | GroupSim | 0.342 | 0.851 | 0.343 |
| MUSCLE | GroupSim | 0.367 | 0.855 | 0.365 |
| TM-Coffee | GroupSim | 0.367 | 0.857 | 0.380 |
| ClustalW | SpeerServer | 0.325 | 0.858 | 0.393 |
| AQUA | GroupSim | 0.392 | 0.863 | 0.415 |
| AQUA | SpeerServer | 0.375 | 0.863 | 0.426 |
| TM-Coffee | SpeerServer | 0.375 | 0.864 | 0.437 |
| MAFFT | GroupSim | 0.400 | 0.867 | 0.435 |
| MUSCLE | SpeerServer | 0.400 | 0.867 | 0.444 |
| MAFFT | SpeerServer | 0.400 | 0.875 | 0.489 |
| ClustalOmega | Xdet | 0.475 | 0.881 | 0.506 |
| TM-Coffee | Xdet | 0.608 | 0.914 | 0.686 |
| T-Coffee | Xdet | 0.542 | 0.912 | 0.716 |
| T-Coffee | - | 0.567 | 0.913 | 0.702 |
| TM-Coffee | - | 0.650 | 0.923 | 0.721 |
| MAFFT | Xdet | 0.642 | 0.925 | 0.745 |
| ClustalOmega | TCS | 0.658 | 0.927 | 0.752 |
| AQUA | Xdet | 0.658 | 0.929 | 0.761 |
| ClustalW | - | 0.675 | 0.931 | 0.767 |
| MUSCLE | - | 0.675 | 0.931 | 0.767 |
| ClustalOmega | - | 0.683 | 0.932 | 0.768 |
| MAFFT | - | 0.683 | 0.932 | 0.769 |
| MUSCLE | Xdet | 0.658 | 0.930 | 0.770 |
| TM-Coffee | TCS | 0.675 | 0.932 | 0.777 |
| AQUA | - | 0.683 | 0.933 | 0.780 |
| ClustalW | Xdet | 0.658 | 0.932 | 0.783 |
| ClustalW | TCS | 0.667 | 0.933 | 0.785 |
| MAFFT | TCS | 0.675 | 0.933 | 0.787 |
| AQUA | TCS | 0.683 | 0.936 | 0.799 |
| MUSCLE | TCS | **0.692** | **0.937** | 0.802 |
| - | - | 0.650 | 0.933 | **0.812** |

The table compares performance of each combination on the 120 sequences of the test set of Dataset One. For each combination the table shows overall accuracy, Macro accuracy, and Overall MCC. They are ranked by their Overall MCC.

Table 4b: Detailed Comparative Performance on Dataset One

| MSA | SDS | Correct Classifications | Misclassifieds | Unclassified |
|---|---|---|---|---|
| T-Coffee | TCS | 4 | 13 | 103 |
| T-Coffee | SpeerServer | 28 | 46 | 46 |
| T-Coffee | GroupSim | 30 | 41 | 49 |
| ClustalOmega | SpeerServer | 37 | 42 | 41 |
| ClustalOmega | GroupSim | 41 | 46 | 33 |
| ClustalW | GroupSim | 41 | 46 | 33 |
| MUSCLE | GroupSim | 44 | 46 | 30 |
| TM-Coffee | GroupSim | 44 | 44 | 32 |
| ClustalW | SpeerServer | 39 | 36 | 45 |
| AQUA | GroupSim | 47 | 42 | 31 |
| AQUA | SpeerServer | 45 | 38 | 37 |
| TM-Coffee | SpeerServer | 45 | 37 | 38 |
| MAFFT | GroupSim | 48 | 40 | 32 |
| MUSCLE | SpeerServer | 48 | 38 | 34 |
| MAFFT | SpeerServer | 48 | 33 | 39 |
| ClustalOmega | Xdet | 57 | 39 | 24 |
| TM-Coffee | Xdet | 73 | 25 | 22 |
| T-Coffee | Xdet | 65 | 19 | 36 |
| T-Coffee | - | 68 | 21 | 31 |
| TM-Coffee | - | 78 | 23 | 19 |
| MAFFT | Xdet | 77 | 20 | 23 |
| ClustalOmega | TCS | 79 | 20 | 21 |
| AQUA | Xdet | 79 | 19 | 22 |
| ClustalW | - | 81 | 19 | 20 |
| MUSCLE | - | 81 | 19 | 20 |
| ClustalOmega | - | 82 | 19 | 19 |
| MAFFT | - | 82 | 19 | 19 |
| MUSCLE | Xdet | 79 | 18 | 23 |
| TM-Coffee | TCS | 81 | 18 | 21 |
| AQUA | - | 82 | 18 | 20 |
| ClustalW | Xdet | 79 | 18 | 23 |
| ClustalW | TCS | 80 | 18 | 22 |
| MAFFT | TCS | 81 | 17 | 22 |
| AQUA | TCS | 82 | 16 | 22 |
| MUSCLE | TCS | 83 | 16 | 21 |
| - | - | 78 | 14 | 28 |

The table compares the detailed performance of each combination on the 120 sequences of the test set of Dataset One. For each combination the table shows number of sequences correctly classified, number of sequences that were missclassified, and the number of unclassified sequences. They are ranked by their Overall MCC.

Table 5a: Comparative Performance on Dataset Two

| MSA | SDS | Micro Accuracy | Macro Accuracy | Overall MCC |
|---|---|---|---|---|
| T-Coffee | TCS | 0.135 | 0.980 | -0.086 |
| T-Coffee | GroupSim | 0.277 | 0.908 | 0.096 |
| T-Coffee | SpeerServer | 0.291 | 0.913 | 0.157 |
| ClustalOmega | GroupSim | 0.365 | 0.912 | 0.190 |
| TM-Coffee | GroupSim | 0.365 | 0.913 | 0.201 |
| MUSCLE | SpeerServer | 0.338 | 0.913 | 0.209 |
| AQUA | SpeerServer | 0.338 | 0.913 | 0.216 |
| ClustalW | GroupSim | 0.378 | 0.913 | 0.224 |
| TM-Coffee | SpeerServer | 0.351 | 0.913 | 0.227 |
| AQUA | GroupSim | 0.372 | 0.912 | 0.228 |
| MUSCLE | GroupSim | 0.378 | 0.915 | 0.231 |
| MAFFT | SpeerServer | 0.338 | 0.912 | 0.232 |
| ClustalW | SpeerServer | 0.358 | 0.919 | 0.243 |
| MAFFT | GroupSim | 0.372 | 0.909 | 0.244 |
| ClustalOmega | Xdet | 0.466 | 0.916 | 0.364 |
| ClustalOmega | SpeerServer | 0.372 | 0.923 | 0.268 |
| TM-Coffee | Xdet | 0.507 | 0.923 | 0.418 |
| ClustalW | - | 0.554 | 0.931 | 0.481 |
| AQUA | Xdet | 0.541 | 0.931 | 0.482 |
| ClustalW | Xdet | 0.554 | 0.932 | 0.488 |
| MAFFT | Xdet | 0.568 | 0.934 | 0.502 |
| ClustalW | TCS | 0.568 | 0.936 | 0.509 |
| AQUA | - | 0.574 | 0.936 | 0.513 |
| AQUA | TCS | 0.581 | 0.939 | 0.533 |
| TM-Coffee | TCS | 0.588 | 0.940 | 0.542 |
| T-Coffee | Xdet | 0.534 | 0.947 | 0.555 |
| MUSCLE | TCS | 0.601 | 0.942 | 0.556 |
| MAFFT | TCS | 0.601 | 0.942 | 0.558 |
| T-Coffee | - | 0.574 | 0.948 | 0.572 |
| ClustalOmega | - | 0.628 | 0.946 | 0.587 |
| TM-Coffee | - | 0.635 | 0.946 | 0.592 |
| MAFFT | - | 0.642 | 0.946 | 0.593 |
| MUSCLE | - | 0.635 | 0.947 | 0.596 |
| MUSCLE | Xdet | 0.635 | 0.947 | 0.599 |
| ClustalOmega | TCS | 0.662 | 0.953 | 0.640 |
| - | - | **0.770** | **0.973** | **0.791** |

The table compares performance of each combination on the 148 sequences of the test set of Dataset Two. For each combination the table shows overall accuracy, Macro accuracy, and Overall MCC. They are ranked by their MCC.

Table 5b: Detailed Comparative Performance on Dataset Two

| MSA | SDS | Correct Classifications | Misclassifieds | Unclassified |
|---|---|---|---|---|
| T-Coffee | TCS | 20 | 16 | 112 |
| T-Coffee | GroupSim | 41 | 75 | 32 |
| T-Coffee | SpeerServer | 43 | 71 | 34 |
| ClustalOmega | GroupSim | 54 | 72 | 22 |
| TM-Coffee | GroupSim | 54 | 71 | 23 |
| MUSCLE | SpeerServer | 50 | 71 | 27 |
| AQUA | SpeerServer | 50 | 71 | 27 |
| ClustalW | GroupSim | 56 | 71 | 21 |
| TM-Coffee | SpeerServer | 52 | 71 | 25 |
| AQUA | GroupSim | 55 | 72 | 21 |
| MUSCLE | GroupSim | 56 | 69 | 23 |
| MAFFT | SpeerServer | 50 | 72 | 26 |
| ClustalW | SpeerServer | 53 | 66 | 29 |
| MAFFT | GroupSim | 55 | 74 | 19 |
| ClustalOmega | Xdet | 69 | 68 | 11 |
| ClustalOmega | SpeerServer | 55 | 63 | 30 |
| TM-Coffee | Xdet | 75 | 63 | 10 |
| ClustalW | - | 82 | 56 | 10 |
| AQUA | Xdet | 80 | 56 | 12 |
| ClustalW | Xdet | 82 | 55 | 11 |
| MAFFT | Xdet | 84 | 54 | 10 |
| ClustalW | TCS | 84 | 52 | 12 |
| AQUA | - | 85 | 52 | 11 |
| AQUA | TCS | 86 | 50 | 12 |
| TM-Coffee | TCS | 87 | 49 | 12 |
| T-Coffee | Xdet | 79 | 43 | 26 |
| MUSCLE | TCS | 89 | 47 | 12 |
| MAFFT | TCS | 89 | 47 | 12 |
| T-Coffee | - | 85 | 42 | 21 |
| ClustalOmega | - | 93 | 44 | 11 |
| TM-Coffee | - | 94 | 44 | 10 |
| MAFFT | - | 95 | 44 | 9 |
| MUSCLE | - | 94 | 43 | 11 |
| MUSCLE | Xdet | 94 | 43 | 11 |
| ClustalOmega | TCS | 98 | 38 | 12 |
| - | - | 114 | 22 | 12 |

The table compares performance of each combination on the 148 sequences of the test set of Dataset Two. For each combination the table shows number of sequences correctly classified, number of sequences that were missclassified, and the number of unclassified sequences They are ranked by their MCC.

Table 6: Performance of *TportHMM* on Dataset One

| Class | Sensitivity | Specificity | Accuracy | MCC | Unclassified |
|---|---|---|---|---|---|
| AminoAcid | 0.73 | 0.99 | 0.96 | 0.80 | 0.07 |
| Anion | 0.50 | 0.96 | 0.92 | 0.50 | 0.08 |
| Cation | 0.81 | 0.98 | 0.93 | 0.82 | 0.19 |
| Electron | 0.40 | 1.00 | 0.95 | 0.62 | 0.60 |
| Protein | 0.47 | 1.00 | 0.93 | 0.66 | 0.53 |
| Sugar | 0.75 | 0.97 | 0.95 | 0.72 | 0.17 |
| Other | 0.65 | 0.98 | 0.93 | 0.71 | 0.23 |
| Overall | | | 0.65 | 0.81 | 0.23 |
| Macro-Average | | | 0.93 | 0.71 | 0.23 |

The table shows performance by class of the *TportHMM* on the 120 sequences of the test set of Dataset One. The column *Unclassified* represents the proportion of sequences that had no match to any profile HMM.

### 3.3.3 Parameters

For BLAST, we used the e-value 0.001 and 120 as the maximum number of hits. Speer Server uses three parameters: *evolutionary rate* (wERate), *relative entropy* (wRE) and *evolutionary distance* (wDist) which are all set to the default value. There are two parameters in GroupSim: the substitution matrix, and the conservation window size. The substitution matrix used is Blosum60 and the window size used is 1. Xdet uses two parameters: the substitution matrix, and the number of random alignments. The random alignments are used to determine the p-value and z-score for correlation. We use *Maxhom_McLachlan* and 10 as the substitution matrix and the number of random alignments respectively. Afterwards, we filter out the columns that have either correlation value of -1 or entropy of -1. *hmmscan* uses the e-value of 0.01.

### 3.3.4 Number of Sequences returned from BLAST

Table 10 shows the number of sequences returned by BLAST in the Dataset One and the Dataset Two. Table 11 and Table 12 shows the sequences which returning zero BLAST hits in the second training set respectively.

Table 7: Confusion Matrix for *TportHMM* on Dataset One

| Actual \ classified | Amino Acid | Anion | Cation | Electron | Protein | Sugar | Other | Unc. |
|---|---|---|---|---|---|---|---|---|
| AminoAcid | 11 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| Anion | 1 | 6 | 1 | 0 | 0 | 1 | 2 | 1 |
| Cation | 0 | 0 | 29 | 0 | 0 | 0 | 0 | 7 |
| Electron | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 6 |
| Protein | 0 | 0 | 0 | 0 | 7 | 0 | 0 | 8 |
| Sugar | 0 | 0 | 0 | 0 | 0 | 9 | 1 | 2 |
| Other | 0 | 4 | 0 | 0 | 0 | 1 | 12 | 3 |

The table shows the confusion matrix of the *TportHMM* on the 120 sequences of the test set of Dataset One. The column *Unc.* represents the number of sequences that had no match to any profile HMM.

### 3.3.5 Number of Subfamilies in each MSA

Table 13 shows the number of subfamilies in each MSA file from MAFFT alignment. Nearly 20% of the first training dataset and 24% of the second training dataset have only one families, therefore they are disregarded for SpeerServer and GroupSim.

## 3.4 Discussion

We are guided by the performance on both datasets to define TportHMM as the primary baseline where the profile HMM is built from a single sequence in the training set. This gives the best overall MCC in Table 4a and Table 5a, hence better performance than the state-of-the-art on Dataset One.

For Dataset One, Table 4a shows the performance for the combination of MSA and SDS tools. The best performing combination is MUSCLE-TCS with an accuracy of 93.7% and overall MCC of 0.802. The primary baseline without any MSA or SDS method has accuracy of 93.3% and overall MCC of 0.812.

For Dataset Two, Table 5a shows the performance of the combinations of MSA and SDS tools. The best performing combination is ClustalOmega-TCS with an accuracy of 95.3% and MCC of 0.640. The primary baseline without any MSA or SDS method has accuracy of 97.3% and MCC of 0.791.

Table 8: Performance of *TportHMM* on Dataset Two

| Class | Sensitivity | Specificity | Accuracy | MCC | Unc. |
|---|---|---|---|---|---|
| Nonselective | 0.50 | 0.99 | 0.98 | 0.40 | 0.00 |
| Water | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 |
| Inorganic Cations | 0.95 | 0.96 | 0.95 | 0.90 | 0.05 |
| Inorganic anions | 0.78 | 0.99 | 0.97 | 0.76 | 0.11 |
| Organic anions | 0.80 | 0.99 | 0.97 | 0.79 | 0.00 |
| Organo-oxygens | 0.76 | 0.97 | 0.95 | 0.73 | 0.00 |
| Amino acids etc | 0.92 | 0.99 | 0.98 | 0.87 | 0.25 |
| Other organonitrogens | 0.57 | 0.98 | 0.94 | 0.61 | 0.14 |
| Nucleotides | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 |
| Organo heterocyclics | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 |
| Miscellaneous | 0.63 | 0.98 | 0.97 | 0.60 | 0.38 |
| Overall | | | 0.77 | 0.79 | 0.09 |
| Macro-Average | | | 0.97 | 0.82 | 0.09 |

The table shows performance by class of the *TportHMM*TportHMM on the 148 sequences of the test set of Dataset Two. The column *Unc.* represents the number of sequences that had no match to any profile HMM.

The impact of the SDS methods when compared to the MSA-baselines without any SDS method is generally weak or negative. This is contrary to the impact seen with SVM classifiers in our previous work [AAB20].

Even the relatively good performance of the primary baseline is surprising, as it indicates a negative impact from using MSA methods.

We expected combinations with TM-Coffee to outperform the other MSA tools, because TM-Coffee is specifically designed to produce alignments consistent with the transmembrane segments, but this was not the case.

### 3.4.1 Comparison with State of the Art

Table 3.4.1 and Figure 10 compares the performance of *TportHMM* to the state-of-the-art *TrSSP* [MCZ14] and *TranCEP* [AAB20] on the independent test set of size 120 from Dataset One. Table 3.4.1 and Figure 11 compares the performance of *TportHMM* to *TrSSP* and *TranCEP* [Alb20]. on the independent test set of size 148 from Dataset Two.

Table 9: Confusion Matrix of *TportHMM* on Dataset Two

| classified / Actual | non-selective | Water | Inorganic Cations | Inorganic anions | Organic anions | Organo-oxygens | Amino acids etc | Other organonitrogens | Nucleotides | Organo heterocyclics | Miscellaneous | Unclassified |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nonselective | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Water | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Inorganic Cations | 2 | 0 | 54 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 3 |
| Inorganic anions | 0 | 0 | 0 | 7 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Organic anions | 0 | 0 | 0 | 1 | 8 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Organo-oxygens | 0 | 0 | 1 | 1 | 0 | 13 | 1 | 1 | 0 | 0 | 0 | 0 |
| Amino acids etc | 0 | 0 | 0 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 1 | 3 |
| Other organonitrogens | 0 | 0 | 2 | 0 | 0 | 3 | 0 | 8 | 0 | 0 | 1 | 2 |
| Nucleotides | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| Organo heterocyclics | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 |
| Miscellaneous | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2 | 0 | 0 | 5 | 3 |

The table shows the confusion matrix of the *TportHMM* on the 148 sequences of the test set of Dataset Two. The column *Unclassified* represents the number of sequences that had no match to any profile HMM.

Table 10: Number of Sequences returned by BLAST

| Number of BLAST hits | Number of sequences in the first training dataset | Number of sequences in the second training dataset |
|---|---|---|
| 111-120 | 10 | 23 |
| 101-110 | 25 | 33 |
| 91-100 | 25 | 19 |
| 81-90 | 26 | 53 |
| 71-80 | 21 | 46 |
| 61-70 | 58 | 54 |
| 51-60 | 32 | 85 |
| 41-50 | 30 | 103 |
| 31-40 | 47 | 142 |
| 21-30 | 61 | 121 |
| 11-20 | 80 | 183 |
| 1-10 | 90 | 200 |

Table 11: Sequences which return zero BLAST Hits in Dataset One

| Substrate class | Sequence ID |
|---|---|
| Amino Acid | P46133 |
| Protein/mRNA | P39742 |
| Other | A2RKV5 |

Table 12: Sequences which return zero BLAST Hits in Dataset Two

| Substrate class | Sequence ID |
|---|---|
| Inorganic Cations | Q12078 |
| | Q6K3T2 |
| | P98161 |
| | O15431 |
| | P69380 |
| | Q03829 |
| | Q18120 |
| | Q9FY75 |
| | P33650 |
| | Q5VT99 |
| | P38054 |
| Inorganic anions | A6YCJ2 |
| | Q9NQ90 |
| Organic anions | P23622 |
| | Q10495 |
| Organo-oxygens | Q9FE59 |
| Other organonitrogens | P69428 |
| | Q8BIJ7 |
| | Q9D1N2 |
| Organo heterocyclics | P39618 |
| Miscellaneous | Q9D232 |
| | O74899 |
| | Q08777 |

| Number of families | Number of MSA in the first training dataset | Number of MSA in the second training dataset |
|---|---|---|
| 1 | 149 | 331 |
| 2 | 145 | 213 |
| 3 | 132 | 246 |
| 4 | 126 | 217 |
| 5 | 103 | 119 |
| 6 | 52 | 91 |
| 7 | 33 | 36 |
| 8 | 19 | 27 |
| 9 | 17 | 60 |
| 11 | 0 | 1 |
| 13 | 0 | 1 |
| 14 | 0 | 1 |
| 16 | 0 | 1 |
| 38 | 0 | 1 |
| 45 | 0 | 1 |
| 46 | 0 | 1 |

Table 13: The number of families returned by Secator in the Dataset One and Dataset Two

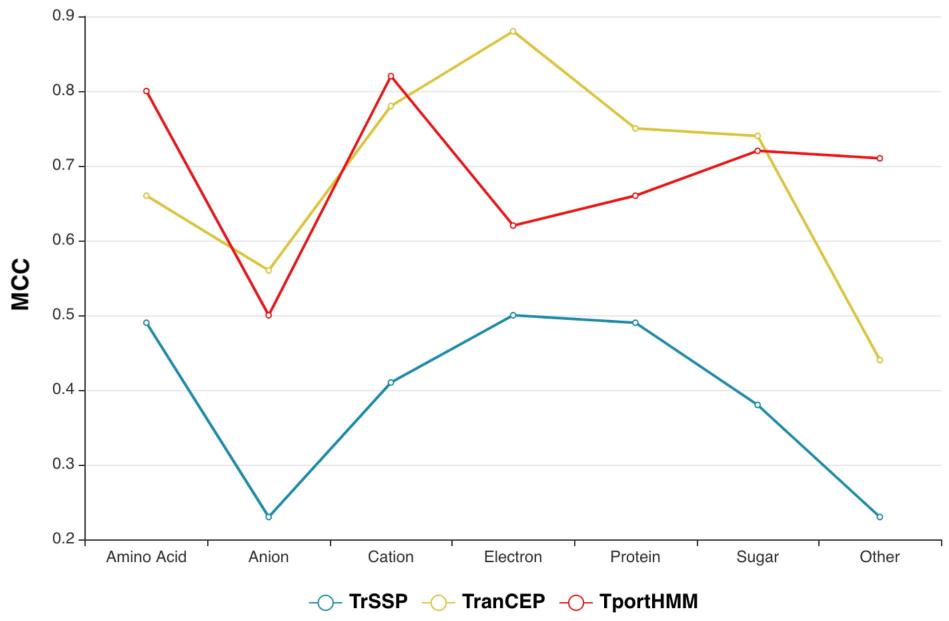


Figure 10: Comparison of *TportHMM* and state-of-the-art on Dataset One

Comparison between *TportHMM* and the state-of-the-art methods (Dataset One).

| Class | Specificity | | | Sensitivity | | | Accuracy | | | MCC | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TrSSP | TranCEP | Tport HMM | TrSSP | TranCEP | Tport HMM | TrSSP | TranCEP | Tport HMM | TrSSP | TranCEP | Tport HMM |
| Amino acid | 82.42 | 96.10 | 99.05 | 93.33 | 93.33 | 73.33 | 83.33 | 91.75 | 95.83 | 0.49 | 0.66 | 0.80 |
| Anion | 69.05 | 96.30 | 96.30 | 75.00 | 58.33 | 50.00 | 69.44 | 90.82 | 91.67 | 0.23 | 0.56 | 0.50 |
| Cation | 74.31 | 89.29 | 97.62 | 75.00 | 58.33 | 80.56 | 74.44 | 89.00 | 92.50 | 0.41 | 0.78 | 0.82 |
| Electron | 91.78 | 100.00 | 100.00 | 80.00 | 80.00 | 40.00 | 91.11 | 97.80 | 95.00 | 0.50 | 0.88 | 0.62 |
| Protein | 82.42 | 99.07 | 100.00 | 93.33 | 66.67 | 46.67 | 83.33 | 93.68 | 93.33 | 0.49 | 0.75 | 0.66 |
| Sugar | 76.79 | 99.07 | 97.22 | 91.67 | 66.67 | 75.00 | 77.78 | 94.68 | 95.00 | 0.38 | 0.74 | 0.72 |
| Other | 73.13 | 86.00 | 96.00 | 60.00 | 65.00 | 60.00 | 71.67 | 80.91 | 90.00 | 0.23 | 0.44 | 0.62 |
| Overall | | | | | | | NA | 74.17 | 65.00 | NA | 0.69 | 0.81 |
| Macro-average | | | | | | | 78.88 | 91.23 | 93.33 | 0.41 | 0.69 | 0.71 |

The table presents our data for TportHMM built with the training set and run on the test set. The corresponding results for TrSSP and TranCEP are taken from their original paper. The table shows specificity, sensitivity, accuracy and MCC for each of the seven substrate types; the overall accuracy and MCC and the average accuracy and MCC. We calculated the overall accuracy as proportion of correct predictions divided by the total number of predictions, and the overall MCC from the confusion matrix as in Equation 11. The TranCEP results were calculated as both the average and overall, while the TrSSP results were calculated only as the average across the seven classes; if we adopt the same methods, we obtain the overall accuracy of 65.00% and overall MCC of 0.81 with the average accuracy is 93.33% and the average MCC is 0.71.

Comparison between *TportHMM* and the state-of-the-art methods (Dataset Two).

| Class | Specificity | | | Sensitivity | | | Accuracy | | | MCC | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TrSSP | TranCEP | Tport HMM | TrSSP | TranCEP | Tport HMM | TrSSP | TranCEP | Tport HMM | TrSSP | TranCEP | Tport HMM |
| C1 | 100.00 | 100.00 | 98.63 | 0.00 | 50.00 | 50.00 | 98.18 | 99.22 | 97.97 | 0.00 | 0.70 | 0.40 |
| C2 | 99.32 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 99.08 | 100.00 | 100.00 | 0.81 | 1.00 | 1.00 |
| C3 | 80.68 | 88.64 | 95.65 | 91.67 | 96.67 | 94.73 | 83.08 | 91.37 | 95.27 | 0.68 | 0.83 | 0.90 |
| C4 | 98.55 | 100.00 | 98.56 | 60.00 | 70.00 | 77.77 | 94.74 | 97.69 | 97.30 | 0.64 | 0.83 | 0.76 |
| C5 | 98.55 | 97.83 | 98.55 | 80.00 | 90.00 | 80.00 | 96.43 | 96.95 | 97.30 | 0.78 | 0.81 | 0.79 |
| C6 | 96.95 | 97.71 | 96.95 | 64.71 | 76.47 | 76.47 | 91.53 | 94.78 | 94.60 | 0.64 | 0.76 | 0.73 |
| C7 | 97.74 | 100.00 | 98.52 | 73.33 | 86.67 | 91.67 | 93.91 | 98.45 | 97.97 | 0.72 | 0.92 | 0.87 |
| C8 | 94.70 | 96.21 | 97.76 | 56.25 | 87.50 | 57.14 | 88.52 | 94.78 | 93.92 | 0.50 | 0.77 | 0.61 |
| C9 | 99.32 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 99.08 | 100.00 | 100.00 | 0.81 | 1.00 | 1.00 |
| C10 | 98.62 | 100.00 | 100.00 | 33.33 | 100.00 | 100.00 | 96.43 | 100.00 | 100.00 | 0.31 | 1.00 | 1.00 |
| C11 | 99.27 | 100.00 | 97.86 | 27.27 | 45.45 | 62.50 | 92.31 | 95.49 | 95.95 | 0.42 | 0.66 | 0.60 |
| Overall | | | | | | | 72.97 | 85.81 | 77.03 | 0.65 | 0.82 | 0.79 |
| Macro-average | | | | | | | 93.94 | 97.16 | 97.30 | 0.57 | 0.84 | 0.82 |

The table presents our data for *TportHMM* built with the training set and run on the test set. The corresponding results for *TrSSP* and *TranCEP* trained and tested with the same dataset. The table shows specificity, sensitivity, accuracy and MCC for each of the eleven substrate types; the overall accuracy and MCC; and the macroaverage accuracy and MCC. The overall accuracy was calculated as the number of correct predictions divided by the total number of predictions, and the overall MCC was calculated from the confusion matrix.

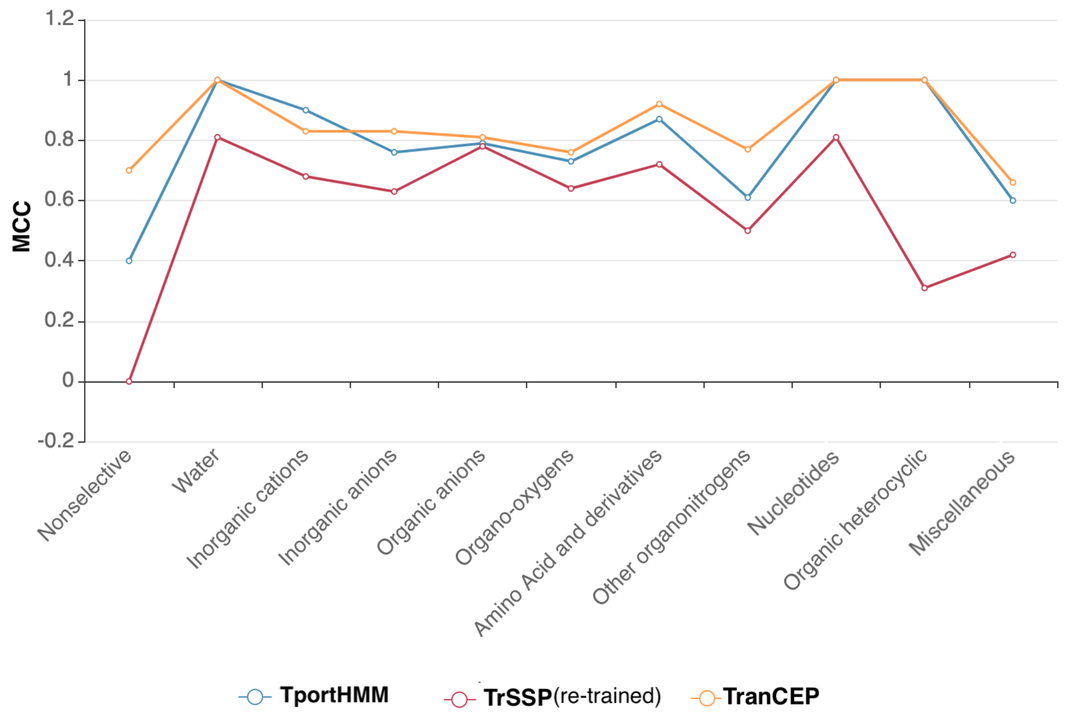


Figure 11: Comparison of *TportHMM* and state-of-the-art on Dataset Two

### 3.4.2 Unclassified Sequences

There are a number of sequences that were not classified by any combinations in both datasets. Table 15 and Table 14 shows the sequences that were unclassified by most of the sequences. Table 10 shows that roughly 12% of training sequences in Dataset One and 16% of training sequences in Dataset Two have less than the 10 hits used as our threshold, indicating that they have few similar sequences in Swiss-Prot, so perhaps no remote homology to the test sequences.

For the performance of GroupSim and SpeerServer, the low result are perhaps due to the high number of MSAs containing only one subfamily therefore disregarded for Training. This needs further invesigation.

Table 14: Unclassified sequences in Dataset One

| | | |
|---|---|---|
| Amino Acids | Q7TNK0 | 30 |
| Cations | P60678 | 34 |
| | P60698 | 36 |
| | Q99JR1 | 30 |
| | Q3TMP8 | 34 |
| | Q9CPC8 | 32 |
| Electron | P0A8Q0 | 35 |
| | P09152 | 36 |
| | P33599 | 33 |
| | P0AFE0 | 27 |
| | P09193 | 36 |
| | P0A616 | 19 |
| Protein | P40477 | 32 |
| | Q6URK4 | 36 |
| | P52870 | 36 |
| | P33754 | 36 |
| | P0AG99 | 35 |
| | P0ABU9 | 31 |
| | Q7Z412 | 36 |
| Sugar | P39344 | 31 |
| | P71067 | 34 |
| Other | P0AFF4 | 21 |
| | O49929 | 36 |
| | P38206 | 27 |

This table presents the sequences in the Dataset One that most of the combinations couldn't classify them.

Table 15: Unclassified sequences in Dataset Two

| Substrate Class | Uniprot ID | Number of combinations |
|---|---|---|
| Inorganic Cations | Q99766 | 36 |
| | Q3EBY6 | 33 |
| | P76425 | 19 |
| | Q15904 | 36 |
| | P03926 | 36 |
| Inorganic anions | Q06598 | 35 |
| Miscalleneous | P69831 | 35 |
| | Q6X893 | 11 |
| | Q91X85 | 16 |
| | Q9DFS4 | 36 |
| Organo-Oxygens | Q9LF50 | 32 |
| | P42905 | 15 |
| Organic anions | Q1EBV7 | 13 |
| | P37327 | 20 |
| Amino Acid and derivatives | P0AD99 | 35 |
| | O60146 | 17 |
| | P57757 | 18 |
| | Q10227 | 16 |
| Other organonitrogens | Q9BZV2 | 15 |
| | P32839 | 17 |
| | P69423 | 18 |
| | Q9NP61 | 34 |
| | Q96A70 | 36 |
| Nucleotides | P39719 | 17 |

This table presents the sequences in the Dataset Two that most of the combinations couldn't classify them.

# Chapter 4

# Conclusion

The performance of TrSSP on Dataset One is given in [MCZ14] and compared with TranCEP in [AAB20]. TrSSP and TranCEP achieve an MCC of 0.41 and 0.69 respectively. Their performance on Dataset Two is given in [Alb20, Table 30 page 115] where TrSSP and TranCEP achieve an MCC of 0.65 and 0.82 respectively. Our method TportHMM achieved an MCC of 0.812 and 0.791 on Dataset One and Dataset Two respectively.

Somewhat surprisingly, the results of our work do not show a benefit in using MSA or SDS methods in the construction of the HMM. Moreover, the SDS methods offer little benefit, or even negative benefit when combined with the different MSA methods. This deserves further investigation.

## 4.1 Limitation of the Work

Although we have analyzed thirty six combinations of seven multiple sequence alignment (MSA) and four different methods for the prediction of specificity determining sites (SDS), there are still a large number of tools for both multiple sequence alignment (MSA) and prediction of specificity determining sites (SDS) and methods that we have not yet tried. There are many MSA tools to be used but we only worked with the most popular ones. However, we used all the SDS tools that we could find the software to install and work on our systems.

Dataset One covers only seven substrate classes. Dataset Two contains examples from eleven substrate classes. The ultimate goal is to predict twenty seven substrate classes because of 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.

## 4.2 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.
- Build the MSAs from the each substrate class rather than on homologous sequences of each sequence in each substrate.
- Understand poor impact of MSA and SDS.
- Build hybrid predictor: best HMM per substrate class.
- Build ensembles from HMM, SVM, and similarity predictors.
- Tuning the parameters for *hmmscan* since the unclassified sequences are the result of *hmmscan* giving score 0.

# Appendix A

# GitHub

The files related to our work can be found at `https://github.com/shivashamloo/TportHMM`. The list of files are presented in Table A.1.

| Files | Discription |
|---|---|
| main.py | A basic file that just takes input |
| finalpipeline.py | Our implementation of combination of methods in terms of a pipeline. This file takes the input from main.py and creates several objects for different stages of the experiment. |
| division.py | divide each sequence in the test and train in a separate file. |
| making_database.py | This file contains two function; the first one is to remove the sequences of the train and test set from the database, the second one is to build a database compatible with blast using blast. |
| blastrun.py | This file contains finding homologous sequences, reformatting the output and merging the training input sequences to its correspondance homologous group. |

Continued on next page

Table A.1: List of files on GitHub

Table A.1 – continued from previous page

| Files | Discription |
|---|---|
| msa.py | This file contains a class called MSA and it has different functions for calculating the MSA based on the input. The output of this file has the following format: MSA_result |
| sds.py | This file is probably the most complicated one. In addition to the function computing the SDS, it also preprocess the MSA to pass them to SDS tools and filter out the MSAs based on the SDS. The output of this section based on SDS method is SDS_result. |
| hmm.py | This file contains a part of training and testsing. First it builds the profile HMMs with *hmmbuild*, second it presses them with *hmmscan* to be compatible with *hmmscan*. And finally it scans each test sequence against the profile HMMs and keeps an score for each search. The training sequence in the profile with the maximum score is considered to be the in the same substrate class as the test sequence. So the prediction is complete. |
| latex.py | This file only computes some measurements and put them in a latex file and creates a *pdf* file. |
| group_sim_sdp.py | a python program of the GroupSim tool downloaded from `https://compbio.cs.princeton.edu/specificity/` |
| blosum62.bla | BLOSUM62 |
| metrix | a folder containing BLOSUM45 |

# Appendix B

# Dataset One test set

Table B.1: Amino Acid

| 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.2: Anion

| 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.3: Cation

| Uniprot ID | TCDB Family |
|---|---|
| Q8AYS8 | R-GGA-1296052 Ca2+ activated K+ channels<br>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<br>2.A.36.4.3 the monovalent cation:proton antiporter-1 (CPA1) family |

Table B.4: Electron transporter

| 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.5: Protein/mRNA

| 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.6: Sugar

| 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.7: Other transporter

| 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 |

# Appendix C

# Dataset Two test set

Table C.1: Nonselective

| Uniprot ID | TCDB Family |
|---|---|
| P0C0S1 | 1.A.23.2.1 The Small Conductance Mechanosensitive Ion Channel (MscS) Family |
| Q61835 | 1.A.27.1.2 The Phospholemman (PLM) Family |

Table C.2: Water

| Uniprot ID | TCDB Family |
|---|---|
| Q40746 | |
| Q6Z2T3 | 1.A.8.12.2 The Major Intrinsic Protein (MIP) Family |

Table C.3: Inorganic cations

| UniprotID | TCDB family |
|---|---|
| Q9SZY7 | |
| Q8IVB4 | 2.A.36.1.19 The Monovalent Cation:Proton Antiporter-1 (CPA1) Family |
| Q9IAL7 | |

Continued on next page

Table C.3 – continued from previous page

| Uniprot ID | TCDB family |
|---|---|
| Q5U1X7 | |
| Q9EYX5 | 1.A.35.4.1 The CorA Metal Ion Transporter (MIT) Family |
| Q21121 | |
| Q7Z443 | 1.A.5.1.2 The Polycystin Cation Channel (PCC) Family |
| Q9XUC4 | 2.A.5.4.18 The Zinc (Zn<sup>2+</sup>)-Iron (Fe<sup>2+</sup>) Permease (ZIP) Family |
| Q8VHX0 | |
| Q54LY6 | |
| Q1PE15 | 1.A.77.1.2 The Mg<sup>2+</sup>/Ca<sup>2+</sup> Uniporter (MCU) Family |
| P52191 | |
| Q28139 | 2.A.19.4.1 The Ca<sup>2+</sup>:Cation Antiporter (CaCA) Family |
| P47818 | 2.A.89.1.1 The Vacuolar Iron Transporter (VIT) Family |
| Q8S1N1 | |
| P48048 | 1.A.2.1.1 The Inward Rectifier K<sup>+</sup> Channel (IRK-C) Family |
| Q99712 | |
| Q9ZTZ7 | 2.A.37.1.4 The Monovalent Cation:Proton Antiporter-2 (CPA2) Family |
| Q9M0Z3 | 2.A.37.1.6 The Monovalent Cation:Proton Antiporter-2 (CPA2) Family |
| P48549 | 1.A.2.1.12 The Inward Rectifier K<sup>+</sup> Channel (IRK-C) Family |
| Q14B80 | |
| P32798 | 2.A.4.2.1 The Cation Diffusion Facilitator (CDF) Family |
| Q99766 | |
| Q21974 | 1.A.6.2.4 The Epithelial Na<sup>+</sup> Channel (ENaC) Family |

Continued on next page

Table C.3 – continued from previous page

| Uniprot ID | TCDB family |
|---|---|
| Q6DBM8 | |
| P34586 | |
| Q75HP9 | |
| Q8W4E7 | 2.A.100.1.2 The Ferroportin (Fpn) Family |
| Q9YDF8 | 1.A.1.17.1 The Voltage-gated Ion Channel (VIC) Superfamily |
| P61829 | 3.A.2.1.3 The H<sup>+</sup>- or Na<sup>+</sup>-translocating F-type, V-type and A-type ATPase (F-ATPase) Superfamily |
| P41807 | 3.A.2.2.3 The H<sup>+</sup>- or Na<sup>+</sup>-translocating F-type, V-type and A-type ATPase (F-ATPase) Superfamily |
| O70596 | 1.A.2.1.5 The Inward Rectifier K<sup>+</sup> Channel (IRK-C) Family |
| P23968 | 3.A.2.2.3 The H<sup>+</sup>- or Na<sup>+</sup>-translocating F-type, V-type and A-type ATPase (F-ATPase) Superfamily |
| P26235 | 2.A.37.2.1 The Monovalent Cation:Proton Antiporter-2 (CPA2) Family |
| Q8TD43 | 1.A.4.5.4 The Transient Receptor Potential Ca<sup>2+</sup> Channel (TRP-CC) Family |
| Q9WU39 | |
| Q9WU38 | |
| P24612 | 1.A.6.2.2 The Epithelial Na<sup>+</sup> Channel (ENaC) Family |
| P35500 | |
| Q8L636 | |
| Q24278 | 1.A.1.5.18 The Voltage-gated Ion Channel (VIC) Superfamily |
| Q8IWT1 | 8.A.17.2.2 The Na<sup>+</sup> Channel Auxiliary Subunit $\beta$1-$\beta$4 (SCA-$\beta$) Family |
| Q06538 | 1.A.17.5.11 The Calcium-dependent Chloride Channel (Ca-ClC) Family |

Continued on next page

Table C.3 – continued from previous page

| Uniprot ID | TCDB family |
|---|---|
| Q9EPK8 | |
| Q96T55 | 1.A.1.9.10 The Voltage-gated Ion Channel (VIC) Superfamily |
| Q8I4B0 | 1.A.1.2.20 The Voltage-gated Ion Channel (VIC) Superfamily |
| P52192 | |
| O95180 | 1.A.1.11.5 The Voltage-gated Ion Channel (VIC) Superfamily |
| Q3EBY6 | 1.A.87.1.2 The Mechanosensitive Calcium Channel (MCA) Family |
| P07293 | 1.A.1.11.2 The Voltage-gated Ion Channel (VIC) Superfamily |
| Q6DHQ1 | |
| P76425 | 2.A.113.1.1 The Nickel/cobalt Transporter (NicO) Family |
| Q12791 | 1.A.1.3.10 The Voltage-gated Ion Channel (VIC) Superfamily |
| Q86LG1 | |
| Q15904 | 8.A.107.1.1 The V-type ATPase assembly factor, ATP6AP1 (ATP6AP1) Family |
| P03928 | |
| G5EBM5 | |
| Q9ERS1 | |
| Q9H427 | 1.A.1.9.12 The Voltage-gated Ion Channel (VIC) Superfamily |
| Q8W4S4 | |

Table C.4: Inorganic anions

| Uniprot ID | TCDB Family |
|---|---|
| Q9XI23 | |
| O35454 | 2.A.49.3.4 The Chloride Carrier/Channel (ClC) Family |
| P25360 | 2.A.47.2.1 The Divalent Anion:Na<sup>+</sup> Symporter (DASS) Family |
| P12234 | 2.A.29.4.1 The Mitochondrial Carrier (MC) Family |
| P92946 | |
| P37002 | 1.A.43.1.1 The Camphor Resistance or Fluoride Exporter (Fluc) Family |
| Q9M817 | 2.A.17.3.13 The Proton-dependent Oligopeptide Transporter (POT/PTR) Family |
| Q06598 | 2.A.59.1.1 The Arsenical Resistance-3 (ACR3) Family |
| A8J6J0 | 2.A.53.1.8 The Sulfate Permease (SulP) Family |
| Q55461 | |

Table C.5: Organic anions

| Uniprot ID | TCDB Family |
|---|---|
| Q9BXS9 | 2.A.53.2.7 The Sulfate Permease (SulP) Family |
| P32847 | 2.A.31.1.4 The Anion Exchanger (AE) Family |
| P58735 | |
| Q1EBV7 | 2.A.28.2.3 The Bile Acid:Na<sup>+</sup> Symporter (BASS) Family |
| P53311 | 2.A.105.1.1 The Mitochondrial Pyruvate Carrier (MPC) Family |
| Q9SFB0 | 2.A.66.1.24 The Multidrug/Oligosaccharidyl-lipid/Polysaccharide (MOP) Flippase Superfamily |
| Q8LG88 | 2.A.47.1.6 The Divalent Anion:Na<sup>+</sup> Symporter (DASS) Family |
| Q9SJE9 | |
| P33303 | 2.A.29.13.1 The Mitochondrial Carrier (MC) Family |
| P37327 | 1.A.16.4.1 The Formate-Nitrite Transporter (FNT) Family |

Table C.6: Organo-oxygens

| Uniprot ID | TCDB Family |
|---|---|
| Q12520 | 2.A.7.11.6 The Drug/Metabolite Transporter (DMT) Superfamily |
| Q66HX0 | |
| P41036 | 2.A.1.12.1 The Major Facilitator Superfamily (MFS) |
| Q8GY97 | |
| Q9BRV3 | 2.A.123.1.4 The Sweet; PQ-loop; Saliva; MtN3 (Sweet) Family |
| Q8K0H7 | |
| P22732 | 2.A.1.1.13 The Major Facilitator Superfamily (MFS) |
| Q9LF50 | 2.A.84.1.1 The Chloroplast Maltose Exporter (MEX) Family |
| Q84L08 | |
| Q94B65 | |
| P42905 | |
| Q7RTX9 | 2.A.1.13.12 The Major Facilitator Superfamily (MFS) |
| B1AT66 | |
| Q7T384 | 2.A.21.5.4 The Solute:Sodium Symporter (SSS) Family |
| P32386 | 3.A.1.208.12 The ATP-binding Cassette (ABC) Superfamily |
| O95342 | 3.A.1.201.2 The ATP-binding Cassette (ABC) Superfamily |
| Q8ZKQ3 | |

Table C.7: Amino acids and derivatives

| Uniprot ID | TCDB Family |
|---|---|
| P0AD99 | 2.A.26.1.10 The Branched Chain Amino Acid:Cation Symporter (LIVCS) Family |
| O34739 | 2.A.3.8.12 The Amino Acid-Polyamine-Organocation (APC) Family |
| Q9URZ4 | 2.A.3.10.21 The Amino Acid-Polyamine-Organocation (APC) Family |
| P11667 | 2.A.75.1.2 The L-Lysine Exporter (LysE) Family |
| Q9JHE5 | 2.A.18.6.4 The Amino Acid/Auxin Permease (AAAP) Family |
| Q60DN5 | |
| Q04671 | 2.A.45.2.1 The Arsenite-Antimonite (ArsB) Efflux Family |
| O60146 | |
| A0A1D8PNP3 | |
| P43548 | 2.A.3.10.14 The Amino Acid-Polyamine-Organocation (APC) Family |
| Q12235 | 2.A.1.14.20 The Major Facilitator Superfamily (MFS) |
| P57757 | |
| Q04301 | 2.A.1.48.1 The Major Facilitator Superfamily (MFS) |
| Q08AI6 | 2.A.18.6.18 The Amino Acid/Auxin Permease (AAAP) Family |
| Q10227 | |

Table C.8: Other organonitrogens

| Uniprot ID | TCDB Family |
|---|---|
| Q9P5N0 | 3.A.1.208.28 The ATP-binding Cassette (ABC) Superfamily |
| Q02592 | 3.A.1.210.2 The ATP-binding Cassette (ABC) Superfamily |
| Q9NP78 | 3.A.1.209.2 The ATP-binding Cassette (ABC) Superfamily |
| Q9BZV2 | 2.A.48.1.4 The Reduced Folate Carrier (RFC) Family |
| P32839 | 3.A.28.1.1 The AAA-ATPase, Bcs1 (Bcs1) Family |
| Q9T091 | |
| P35818 | 1.B.22.1.2 The Outer Bacterial Membrane Secretin (Secretin) Family |
| P69423 | 2.A.64.1.1 The Twin Arginine Targeting (Tat) Family |
| P35179 | 3.A.5.8.1 The General Secretory Pathway (Sec) Family |
| P32897 | 3.A.8.1.1 The Mitochondrial Protein Translocase (MPT) Family |
| Q8LPR8 | 3.A.9.1.2 The Chloroplast Envelope Protein Translocase (CEPT or Tic-Toc) Family |
| Q9W552 | |
| Q9NP61 | |
| P53134 | 2.A.67.2.7 The Oligopeptide Transporter (OPT) Family |
| P0AGH5 | 3.A.1.5.42 The ATP-binding Cassette (ABC) Superfamily |
| Q96A70 | |

Table C.9: Nucleotides

| Uniprot ID | TCDB Family |
|---|---|
| P39719 | 1.A.4.9.3 The Transient Receptor Potential Ca<sup>2+</sup> Channel (TRP-CC) Family |
| Q20799 | |

Table C.10: Organo heterocyclics

| Uniprot ID | TCDB Family |
|---|---|
| P0AGM9 | 2.A.40.4.2 The Nucleobase/Ascorbate Transporter (NAT) or Nucleobase:Cation Symporter-2 (NCS2) Family |
| Q9LZD0 | 2.A.39.3.11 The Nucleobase:Cation Symporter-1 (NCS1) Family |
| O43868 | 2.A.41.2.4 The Concentrative Nucleoside Transporter (CNT) Family |

Table C.11: Miscellaneous

| Uniprot ID | TCDB Family |
|---|---|
| P69831 | 4.A.5.1.1 The PTS Galactitol (Gat) Family |
| P39352 | |
| Q6X893 | 2.A.92.1.1 The Choline Transporter-like (CTL) Family |
| P0AAG0 | |
| O43000 | 2.A.1.14.17 The Major Facilitator Superfamily (MFS) |
| Q91X85 | |
| Q99PE8 | |
| P16258 | |
| Q9BZF2 | |
| Q9GQQ0 | 2.A.1.49.1 The Major Facilitator Superfamily (MFS) |
| Q9DFS4 | |

# Bibliography

[AAB20] Munira Alballa, Faizah Aplop, and Gregory Butler. TranCEP: Predicting the substrate class of transmembrane transport proteins using composition, evolutionary, and positional information. *PLoS ONE*, 15(1):e0227683, 2020.

[AB19] Munira Alballa and Gregory Butler. Ontology-based transporter substrate annotation for benchmark datasets. In Illhoi Yoo, Jinbo Bi, and Xiaohua Hu, editors, *2019 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2019, San Diego, CA, USA, November 18-21, 2019*, pages 2613–2619. IEEE, 2019.

[AGM$^+$90] 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.

[Alb20] Munira Alballa. *Predicting transporter proteins and their substrate specificity.* PhD thesis, Concordia University, 2020.

[AMS$^+$97] 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.

[BH13] Ahmad Barghash and Volkhard Helms. Transferring functional annotations of membrane transporters on the basis of sequence similarity and sequence motifs. *BMC Bioinformatics*, 14(1):343, 2013.

[BSS$^+$10] 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.

[CBP07] 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.

[CC15] Abhijit Chakraborty and Saikat Chakrabarti. A survey on prediction of specificity-determining sites in proteins. *Briefings in Bioinformatics*, 16(1):71–88, 2015.

[CDTN14] Jia-Ming Chang, Paolo Di Tommaso, and Cedric Notredame. TCS: a new multiple sequence alignment reliability measure to estimate alignment accuracy and improve phylogenetic tree reconstruction. *Molecular Biology and Evolution*, pages 1625—1637, 2014.

[CML$^+$12] Abhijit Chakraborty, Sapan Mandloi, Christopher J. Lanczycki, Anna R. Panchenko, and Saikat Chakrabarti. SPEER-SERVER: a web server for prediction of protein specificity determining sites. *Nucleic Acids Research*, 40(Web Server issue):W242–W248, 2012.

[COLG11] ShuAn Chen, YuYen Ou, TzongYi Lee, and M Michael Gromiha. Prediction of transporter targets using efficient RBF networks with PSSM profiles and biochemical properties. *Bioinformatics*, 27(15):2062–2067, 2011.

[Com20a] Wikimedia Commons. File:blosum62.png — wikimedia commons, the free media repository, 2020. [Online; accessed 9-January-2021].

[Com20b] Wikimedia Commons. File:cell membrane detailed diagram en.svg — wikimedia commons, the free media repository, 2020. [Online; accessed 29-December-2020].

[CS08] John A. Capra and Mona Singh. Characterization and prediction of residues determining protein functional specificity. *Bioinformatics*, 24(13):1473–1480, 2008.

[DOS13] Jurate Daugelaite, Aisling O'Driscoll, and Roy D Sleator. An overview of multiple sequence alignments and cloud computing in bioinformatics. *International Scholarly Research Notices*, 2013:615–630, 2013.

[EB06] Robert C Edgar and Serafim Batzoglou. Multiple sequence alignment. *Current Opinion in Structural Biology*, 16(3):368 – 373, 2006.

[Edd98] S R Eddy. Profile hidden markov models. *Bioinformatics*, 14(9):755–763, 1998.

[Edg04] Robert C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797, 2004.

[FCE11] Robert D Finn, Jody Clements, and Sean R Eddy. HMMER web server: interactive sequence similarity searching. *Nucleic Acids Research*, 39(suppl_2):W29–W37, 2011.

[Fla19] P.M. Flatt. Protein structure. In *Biochemistry – Defining Life at the Molecular Level*, chapter 2. Western Oregon University, 2019.

[FTC$^+$16] 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.

[GO14] MM Gromiha and YY Ou. Bioinformatics approaches for functional annotation of membrane proteins. *Briefings in Bioinformatics*, 15(2):155–168, 2014.

[Gor04] Jan Gorodkin. Comparing two K-category assignments by a K-category correlation coefficient. *Computational Biology and Chemistry*, 28(5):367–374, 2004.

[Gu01] Xun Gu. Maximum-likelihood approach for gene family evolution under functional divergence. *Molecular Biology and Evolution*, 18(4):453–464, 2001.

[GY08] M Michael Gromiha and Yukimitsu Yabuki. Functional discrimination of membrane proteins using machine learning techniques. *BMC Bioinformatics*, 9(1):135, 2008.

[HH92] 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.

[HM91] Xiaoqiu Huang and Webb Miller. A time-efficient, linear-space local similarity algorithm. *Advances in Applied Mathematics*, 12(3):337 – 357, 1991.

[KBM$^+$94] 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.

[Kim91] Motoo Kimura. The neutral theory of molecular evolution: A review of recent evidence. *The Japanese Journal of Genetics*, 66(4):367–386, 1991.

[KMKM02] Kazutaka Katoh, Kazuharu Misawa, Kei-ichi Kuma, and Takashi Miyata. Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic acids research*, 30(14):3059–3066, 2002.

[LBUZ09] 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(418):1–13, 2009.

[MCT$^+$09] Jean Muller, Christopher J. Creevey, Julie D. Thompson, Detlev Arendt, and Peer Bork. AQUA: automated quality improvement for multiple sequence alignments. *Bioinformatics*, 26(2):263–265, 11 2009.

[MCZ14] 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.

[NHH00] 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.

[Not02] Cedric Notredame. Recent progress in multiple sequence alignment: a survey. *Pharmacogenomics*, 3(1):131–144, 2002.

[Not07] Cedric Notredame. Recent evolutions of multiple sequence alignment algorithms. *PLOS Computational Biology*, 3(8):1–4, 08 2007.

[OCG10] Yu-Yen Ou, Shu-An Chen, and M Michael Gromiha. Classification of transporters using efficient radial basis function networks with position-specific scoring matrices and biochemical properties. *Proteins: Structure, Function, and Bioinformatics*, 78(7):1789–1797, 2010.

[PBM$^+$02] 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.

[Pev09a] Jonathan Pevsner. Molecular phylogeny and evolution. In *Bioinformatics and Functional Genomics*, chapter 7. Wiley-Blackwell, Baltimore, Maryland, 2 edition, 2009.

[Pev09b] Jonathan Pevsner. Multiple sequence alignment. In *Bioinformatics and Functional Genomics*, chapter 6. Wiley-Blackwell, 2009.

[PF01] Alexander Pertsemlidis and John W Fondon. Having a blast with bioinformatics (and avoiding blastphemy). *Genome Biol*, 2(10), 2001.

[PFH08] Walter Pirovano, K. Anton Feenstra, and Jaap Heringa. Praline: a strategy for improved multiple alignment of transmembrane proteins. *Bioinformatics*, 24(4):492–497, 2008.

[PRV06] Florencio Pazos, Antonio Rausell, and Alfonso Valencia. Phylogeny-independent detection of functional residues. *Bioinformatics*, 22(12):1440–1448, 2006.

[SCH10] 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.

[SH12] NS Schaadt and V Helms. Functional classification of membrane transporters and channels based on filtered TM/non-TM amino acid composition. *Biopolymers*, 97(7):558–567, 2012.

[SJRT$^{+}$16] Milton H Saier Jr, Vamsee S Reddy, Brian V Tsu, Muhammad Saad Ahmed, Chun Li, and Gabriel Moreno-Hagelsieb. The transporter classification database (TCDB): recent advances. *Nucleic Acids Research*, 44(D1):D372–D379, 2016.

[SN87] 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.

[SWD$^{+}$11] 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.

[Sö05] Johannes Söding. Protein homology detection by HMM–HMM comparison. *Bioinformatics*, 21(7):951–960, 2005.

[THG94] 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):4673–4680, 1994.

[TPP99] Julie D. Thompson, Frédéric Plewniak, and Olivier Poch. A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Research*, 27(13):2682–2690, 07 1999.

[TPTP01] J. D. Thompson, R. Plewniak, F. Ripp, J. C. Thierry, and O. Poch. Towards a reliable objective function for multiple sequence alignments. *Molecular Biology*, 2001.

[TTP03] J. D. Thompson, J. C. Thierry, and O. Poch. Rascal: rapid scanning and correction of multiple sequence alignments. *Bioinformatics*, 19:1155–1161, 2003.

[TWNB12] Elin Teppa, Angela D. Wilkins, Morten Nielsen, and Cristina Marino Buslje. Disentangling evolutionary signals: conservation, specificity determining positions and coevolution. implication for catalytic residue prediction. *BMC Bioinformatics*, 13:235, 2012.

[WPTP01] Nicolas Wicker, Guy René Perrin, Jean Claude Thierry, and Olivier Poch. Secator: A program for inferring protein subfamilies from phylogenetic trees. *Molecular Biology and Evolution*, 18(8):1435–1441, August 2001.