PREDICTING TRANSPORTER PROTEINS AND THEIR SUBSTRATE SPECIFICITY

Munira Alballa

A Thesis

in

the department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Doctor of Philosophy (Computer Science)
Concordia University
Montréal, Québec, Canada

June, 2020

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By:	Munira Alballa
Entitled:	Predicting transporter proteins and their substrate specificity

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

Doctor of Philosophy (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 committee:

Dr. Ciprian Alecsandru	Chair
Dr. Anthony J. Kusalik	External Examiner
Dr. Leila Kosseim	Examiner
Dr. Adam Krzyzak	Examiner
Dr. Jia Yuan Yu	Examiner
Dr. Gregory Butler	Supervisor

Approved

Dr. Leila Kosseim, Graduate Program Director

June 5, 2020

Dr. Amir Asif, Dean

Faculty of Engineering and Computer Science

Abstract

Predicting transporter proteins and their substrate specificity

Munira Alballa, Ph.D.
Concordia University, 2020

The publication of numerous genome projects has resulted in an abundance of protein sequences, a significant number of which are still unannotated. Membrane proteins such as transporters, receptors, and enzymes are among the least characterized proteins due to their hydrophobic surfaces and lack of conformational stability. This research aims to build a proteome-wide system to determine transporter substrate specificity, which involves three phases: 1) distinguishing membrane proteins, 2) differentiating transporters from other functional types of membrane proteins, and 3) detecting the substrate specificity of the transporters.

To distinguish membrane from non-membrane proteins, we propose a novel tool, *TooT-M*, that combines the predictions from transmembrane topology prediction tools and a selective set of classifiers where protein samples are represented by pseudo position-specific scoring matrix (Pse-PSSM) vectors. The results suggest that the proposed tool outperforms all state-of-the-art methods in terms of the overall accuracy and Matthews correlation coefficient (MCC).

To distinguish transporters from other proteins, we propose an ensemble classifier, *TooT-T*, that is trained to optimally combine the predictions from homology annotation transfer and machine learning methods. The homology annotation transfer components detect transporters by searching against the transporter classification database (TCDB) using different thresholds. The machine learning methods include three models wherein the protein sequences are encoded using a novel encoding *psi-composition*. The results show that *TooT-T* outperforms all state-of-the-art *de novo* transporter predictors in terms of the overall accuracy and MCC.

To detect the substrate specificity of a transporter, we propose a novel tool, *TooT-SC*, that combines compositional, evolutionary, and positional information to represent protein samples. *TooT-SC* can efficiently classify transport proteins into eleven classes according to their transported substrate, which is the highest number of predicted substrates offered by any *de novo* prediction tool. Our results indicate that *TooT-SC* significantly outperforms all of the state-of-the-art methods. Further analysis of the locations of the informative positions reveals that there are more statistically significant informative positions in the transmembrane segments (TMSs) than the non-TMSs, and there are more statistically significant informative positions that occur close to the TMSs compared to regions far from them.

Acknowledgments

First and foremost, I would like to thank Almighty Allah for His guidance, mercy and grace throughout my life and during this journey.

My deepest gratitude goes to my academic supervisor, Dr. Gregory Butler, for his time, valuable advice, and encouragement during this research project. I have been extremely lucky to work with a supervisor who cared about my work and my well-being as much as he did. He has demonstrated every aspect of what a good supervisor should be, and I aspire to do the same for my students one day.

I am forever indebted to my parents Suliman and Aljowhara, who instilled in me the value of education. I thank them for believing in me and always having my back. This research would not be possible without their prayers, sacrifice and support. I owe them everything.

I am sincerely and heartily grateful to my siblings: my eldest sister Hailah for her invaluable suggestions on my topic and her expertise, my sister Norah for always being there for me, my younger siblings for their sweetness and humor that helped me go through the tough times.

I would like to thank my partners in this journey, my beloved husband Maan for his encouragement and support and my sweet son Sulaiman for putting up with me being a part-time mommy all these years. Thank you for making my days shine and giving me a reason to smile every day. I love you.

Last but not least, I would like to express my deepest appreciation to King Saud University and the Saudi Cultural Bureau for giving me the opportunity to continue my education and providing me with the financial support.

Contents

List of Figures

List of Tables

Glossary

Cell membrane Biological membrane that surrounds the cytoplasm of living cells, physically separating the intracellular components from the extracellular environment.

ChEBI Chemical Entities of Biological Interest, database and ontology of molecular entities [HOD+15].

DS-M Benchmark dataset of membrane proteins. Collected from the Swiss-Prot database by this research.

DS-SC Benchmark dataset of eleven substrate classes. Collected from the Swiss-Prot database by this research.

DS-T Benchmark dataset of transporter proteins. Collected from the Swiss-Prot database by this research.

Fasttrans Tool developed to predict seven classes of substrate-specific transporter as well as transporter/non-transporter [HPO+19].

GO Gene Ontology, set of three controlled vocabularies (MF, BP, CC) to describe the role of a gene product [ABB+00]

HMMTOP Tool to predict the topology of transmembrane α-helical segments [TS01]. Available online: `http://www.enzim.hu/hmmtop/html/adv_submit.html`

Homoeostasis Property of a system in which variables are regulated so that internal conditions remain stable and relatively constant.

Homologous The existence of shared ancestry between a pair of structures, or genes, in different species.

Hydrophilic Interacting effectively with water.

Hydrophobic Not interacting effectively with water; in general, poorly soluble or insoluble in water.

iMem-2LSAAC Tool to predict membrane proteins [AHJ18].

MemType-2L Tool to predict membrane proteins [CS07].

Nonpolar A molecule or structure that lacks any net electric charge or asymmetric distribution of positive and negative charges. Nonpolar molecules generally are insoluble in water.

Ontoclass Tool developed by this research to assign a substrate class label to any given transporter using existing databases and ontologies.

Polar A Molecule or structure with a net electric charge or asymmetric distribution of positive and negative charges. Polar molecules are usually soluble in water.

PRED-TMBB2 Tool to predict the topology of transmembrane β-barrel segments [TEB16]. Available online: `http://www.compgen.org/tools/PRED-TMBB2`.

Protein sequence The unique sequence of amino acids that characterizes a given protein.

psiAAC Method of encoding a protein sequence into a numerical vector, proposed by this research. It combines amino acid composition with evolutionary information obtained from PSI-BLAST.

psiPAAC Method of encoding a protein sequence into a numerical vector, proposed by this research. It combines pair amino acid composition with evolutionary information obtained from PSI-BLAST.

psiPseAAC Method of encoding a protein sequence into a numerical vector, proposed by this research. It combines pseudo-amino acid composition with evolutionary information obtained from PSI-BLAST.

R Programming language and environment for statistical computing and graphics.

SCMMTP Tool to predict transporter proteins [LVY+15].

TC IUBMB approved system of nomenclature for transport protein classification.

TMHMM Tool to predict the topology of transmembrane α-helical segments [KLvHS01]. Available online: `http://www.cbs.dtu.dk/services/TMHMM/`.

TMC-TCS-PAAC Method of encoding a protein sequence into a numerical vector, proposed by this research. It combines pair amino acid composition PAAC with evolutionary information using TM-Coffee, and filters unreliable columns as determined by TCS.

TooT-M Tool developed by this research to distinguish membrane proteins from non-membrane proteins.

TooT-SC Tool developed by this research to predict eleven classes of substrate-specific transporter.

TooT-T Tool developed by this research to distinguish transporter proteins from non-transporter membrane proteins.

TOPCONS2 Tool to predict the topology of transmembrane α-helical segments [TPS+15]. Available online: : `http://topcons.cbr.su.se/`.

TranCEP Tool developed by this research to predict the same seven classes of substrate-specific transporter as TrSSP. It utilizes PAAC-TMC-TCS method and is trained and tested using TrSSP training and testing datasets, respectively.

TrSSP Tool developed by Mishra et al. [MCZ14] to predict seven classes of substrate-specific transporter as well as transporter/non-transporter.

TrSSP dataset Benchmark dataset of seven substrate classes. Collected from the Swiss-Prot database by Mishra and utilized to develop TrSSP [MCZ14]. Available online: `http://bioinfo.noble.org/trssp/?dowhat=datasets`.

Abbreviations

AAC	Amino Acid Composition
ATH	Annotation Transfer by Homology
BLAST	Basic Local Alignment Search Tool
BLOSUM	BLOcks SUbstitution Matrix
BP	Biological Process
BMC	BioMed Central
CC	Cellular Compartment
CV	Cross-Validation
EC	Enzyme Commission
ECO	Evidence and Conclusion Ontology
GBM	Gradient-Boosting Machine
GPCR	G Protein-Coupled Receptors
GPI	Glycosylphosphatidylinositol
HSP	Highest-Scoring Pairs
IEA	Inferred from Electronic Annotation
IMP	Integral Membrane Proteins
IUBMB	International Union of Biochemistry and Molecular Biology
IUPAC	International Union of Pure and Applied Chemistry
KNN	K-Nearest Neighbor
LOOCV	Leave-One-Out Cross-Validation
MCC	Matthews Correlation Coefficient

MF	Molecular Function
MI	Mutual Information
MID	Mutual Information Difference
mRMR	minimum Redundancy Maximum Relevance
MSA	Multiple Sequence Alignment
MSA-AAC	Multiple Sequence Alignment Amino Acid Composition
NC	Nomenclature Committee
NLP	Natural Language Processing
OET-KNN	Optimized Evidence-Theoretic K-Nearest Neighbor
ORF	Open Reading Frame
PAAC	Pair Amino Acid Composition
PAM	Point Accepted Mutations
Pse-PSSM	Pseudo Position-Specific Scoring Matrix
PseAAC	Pseudo-Amino Acid Composition
PSI-BLAST	Position-Specific Iterated BLAST
PSSM	Position-Specific Scoring Matrix
RBF	Radial Basis Function
SAAC	Split Amino Acid Composition
SCM	Scoring Card Method
SVM	Support Vector Machine
TCDB	Transporter Classification Database
TCID	Transport Classification IDentifier
TCS	Transitive Consistency Score
TMC	TM-Coffee
TMS	Transmembrane Segments

Chapter 1

Introduction

1 Biological background

Due to their biological significance, cell membranes are the only cellular structures found in all cells of all organisms on earth. Membranes maintain the integrity of the cell by separating the critical chemicals and structures necessary to protect the cell from the surrounding environment. They serve as gatekeepers, regulating the flow of molecules, energy, and information into and out of the cell. Eukaryotic cells also have internal membranes that enclose their organelles and control the exchange of essential cell components [LBZ+00].

Cell membranes consist of two main components: lipids and proteins (see Figure 1). Each component has clearly defined functions; for example, lipids form the universally conserved bilayer structure, which has basic barrier properties that determine membrane flexibility and how membrane proteins bind to the lipid bilayer. Membrane proteins enable the membrane to perform distinctive activities with a vast diversity of cell membrane functions.

The lipid bilayer consists of two layers of phospholipid molecules whose fatty tails form the hydrophobic interior, and their hydrophilic (polar) heads line both the inside and outside of the cell surface. Membrane proteins are embedded within, or interact with, the lipid bilayer; they adopt different forms based on the cell type and their location.

Some membrane proteins bind only to the membrane surface, others span the entire

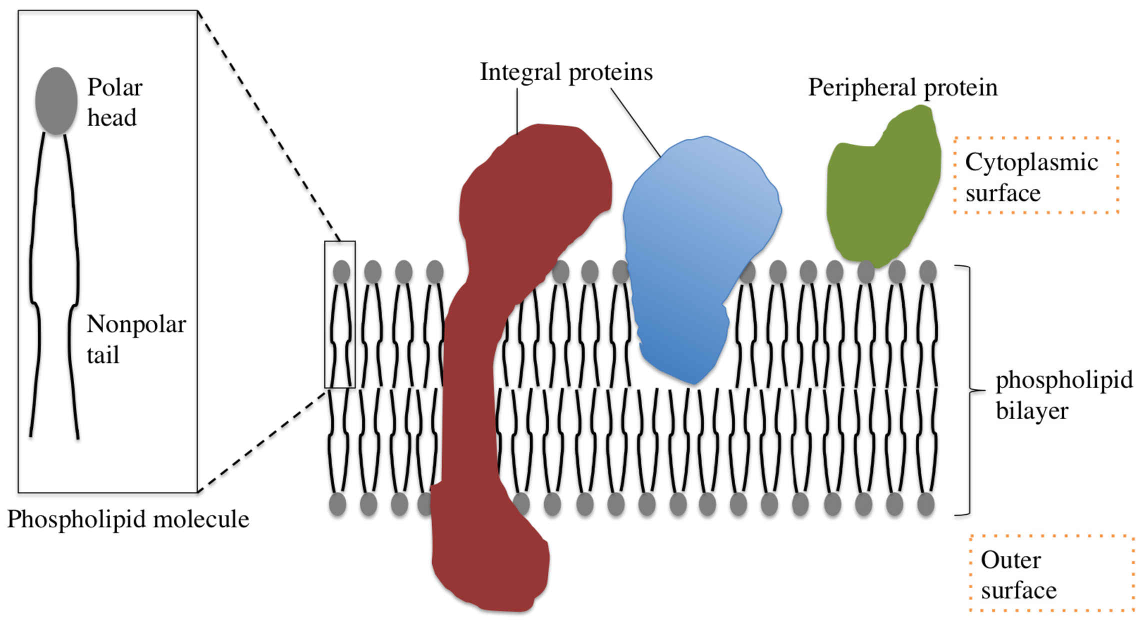


Figure 1: The structure of the cell membrane
This figure illustrates the two main components of the cell membrane: the lipid bilayer and membrane proteins. Membrane proteins can be either surface-bound (e.g., peripheral) or integral.

lipid bilayer and are exposed to water-soluble domains on both sides of the membrane (see Figure 2). The proteins buried within the lipid bilayer are integral membrane proteins (IMPs) (also called "transmembrane proteins"). They have one or more transmembrane segments (TMSs) embedded in the lipid bilayer in addition to extramembranous hydrophilic segments extending into the water-soluble domains on each side of the lipid bilayer. The embedded segments are distinguishable since they contain residues with hydrophobic properties that interact with the hydrophobic (nonpolar) tails inside the membrane phospholipids.

The TMSs in membrane proteins take two general structural forms: α-helix or β-sheet. Integral proteins of the α-helix TMSs are formed by the connection of helices with extramembranous loops, the extramembranous domains could contain both α-helix and β-sheet structures. Integral proteins of the β-sheet TMSs are composed of transmembrane β-strands that stretch across the lipid bilayer and align in an antiparallel fashion into large, self-enclosed β-sheets with extramembranous loops connecting adjacent β-strands. The extramembranous loops generally lack a secondary structure (random coils), but some longer loops may contain exceptionally small α-helical regions [Bue15].

Unlike membrane proteins with α-helix TMSs, which are found in abundance in all cellular membranes [vH99], membrane proteins with β-barrel TMSs are found experimentally only in the outer membranes of gram-negative bacteria. However, some weak similarities at the sequence level indicate that β-barrel membrane proteins may be present in the outer membrane of mitochondria and chloroplasts [Sch03].

Conversely, surface-bound proteins do not expand into the hydrophobic interior of the lipid bilayer; they are typically bound to lipid head groups at the membrane surface or indirectly by attaching to other IMPs. Surface-bound proteins such as peripheral and lipid-anchored proteins do not have the hydrophobic properties of the IMPs; they are therefore more difficult to distinguish than the IMPs.

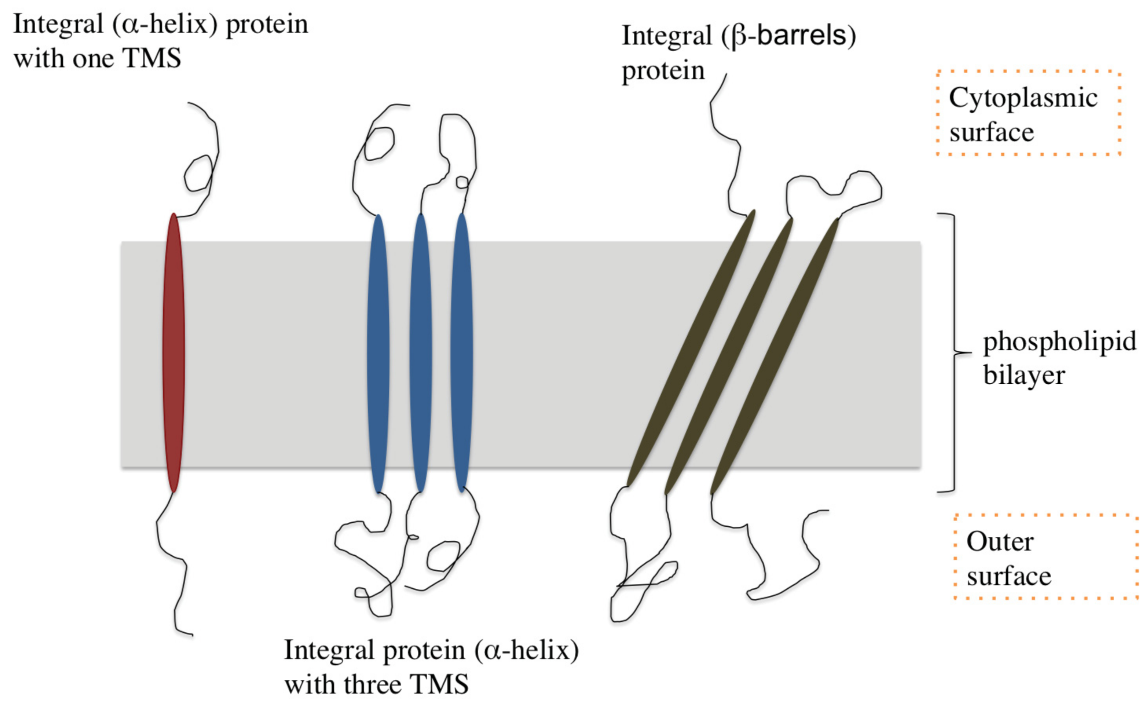

Integral (α-helix) protein with one TMS
Integral (β-barrels) protein
Cytoplasmic surface
phospholipid bilayer
Outer surface
Integral protein (α-helix) with three TMS

Figure 2: Schematic representation of transmembrane proteins
IMPs (transmembrane proteins) have one or more segments embedded within the lipid bilayer (TMSs) connected by extramembranous loops.

Protein structures are described in four distinct levels of hierarchical organization: primary, secondary, tertiary and quaternary structures. These levels denote the amino acid sequence of the protein, the local regular substructures (e.g., α-helices and β-strands), the 3D structure of a single polypeptide, and the aggregation of two or more individual polypeptide chains that comprise the protein complex, respectively. This research specifically examines the primary and secondary structures of proteins.

1.1 Membrane protein functional classes

Membrane proteins control nearly all functions of the membrane, aside from the basic barrier property of the lipid bilayer. The categorization of membrane proteins based on their function was applied long before high-resolution structural methods became available [Bue15]. This makes sense conceptually, because the underlying structure of such proteins can only be identified once their functionality is determined [Bue15]. Membrane proteins can be classified into four different functional groups [Bue15] (Figure 3).

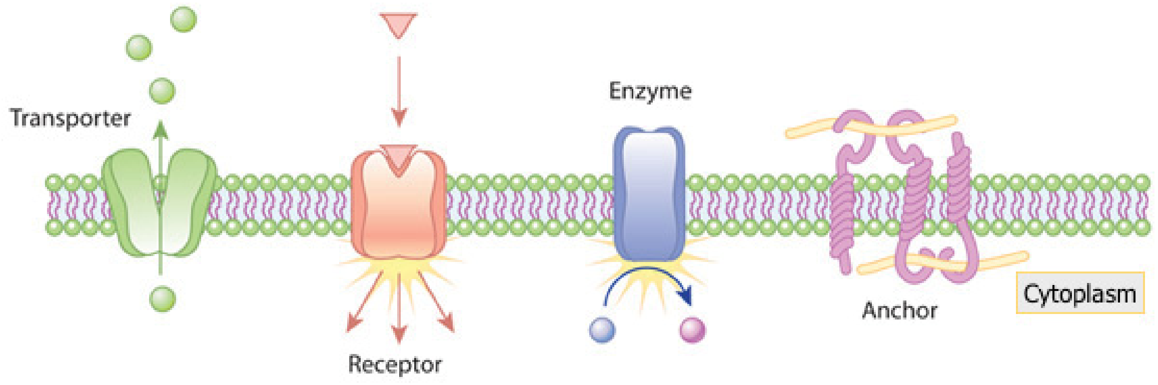


Figure 3: Membrane protein functional classes
This figure shows the four main functional classes of membrane proteins. Transporters allow certain molecules to enter or leave the cell. Receptors can bind an extracellular molecule, which activates an intracellular process. Enzymes can transform a molecule into another form; they perform this function in the cytoplasmic side of the membrane. Anchor proteins physically link intracellular structures with extracellular structures. The figure is from [OAF10].

- Transporters are responsible for selective permeability. They are highly selective, allowing only certain substrates to enter or leave the cell. Channels and carriers are two major types of transporters.

- Receptors are responsible for the binding of extracellular signaling molecules and the generation of various intracellular signals on the opposite side of the plasma membrane.

- Enzymes are responsible for various chemical reactions in the interior surface of the plasma membrane.
- Anchor proteins are responsible for cell adhesion and contain cell surface identity markers.

2 Motivation

Membrane proteins are key gatekeepers that control various vital cellular functions, including cell signaling, trafficking, metabolism, and energy production. It is estimated that one in every three proteins found in a cell of an average organism is a membrane protein. Human genome analysis, for example, predicts that 20% to 30% of all open reading frames (ORFs) encode membrane proteins [WvH98]. Transporters are membrane proteins that control the flow of molecules into and out of the cell; they play critical roles in cellular homeostasis and are attractive targets for the pharmaceutical industry [G$^+$06]. For example, neurotransmitter transporters are the targets of drugs used in the treatment of neuropsychiatric disorders [IKY02], and serotonin transporters are the targets of a major class of antidepressants, serotonin selective reuptake inhibitors [G$^+$06]. In addition, alterations to the function and/or expression of Na^+-dependent glutamate transporters have been implicated in a range of psychiatric and neurological disorders, such as epilepsy, Alzheimer's disease, Huntington's disease, HIV-associated dementia, amyotrophic lateral sclerosis (ALS) and malignant gliomas [SSS04].

As a result of numerous recent genome projects, the sequences of many membrane proteins are now known; however, their structure and function remain poorly characterized and understood because of the immense effort required to characterize them. Generally (for all proteins), experimentally identifying the function of a protein is not a trivial task because the function may be related specifically to the native environment in which a particular organism lives, and such an environment is difficult to simulate in a laboratory. In particular, membrane proteins have a hydrophobic surface, which requires the use of detergents to extract them from the cell membrane. Additionally, their flexibility and instability create challenges at many levels, including crystallization, expression, and structure solution [CBCI08].

The Protein Data Bank (PDB) is an example of how membrane proteins are less represented than other types of soluble proteins; the PDB is the only worldwide repository of data on the 3D structures of large biological molecules, such as proteins and nucleic acids. As of March 2020, less than 4% of the PDB is membrane proteins, of which 2.9% are α-helical structures and 0.6% are β-barrel structures.

Therefore, the detection of membrane proteins and the knowledge of transporters and transport mechanisms are fundamental to the advancement of functional and structural genomics. It is thus extremely desirable to make use of membrane protein sequences, along with the available experimental data in computational tools, to detect membrane proteins and determine their function. Such tools can serve as a guide to diminish the search space for researchers when determining the function of novel proteins. Current state-of-the-art methods remain far from delivering a solution, but initial attempts have been made that require further improvements.

3 Problem definition

Regarding predicting transporters and their functions, not only is finding a solution difficult but so is asking the right question. This is difficult mainly because the concepts related to membrane transport proteins are poorly defined and there is no single coherent problem for predicting a transport protein that all methods agree upon. Rather, there are different perspectives on different levels of prediction.

Even gold standard databases are not consistent. The Transporter Classification Database (TCDB) [LBUZ09], which offers the gold standard classification for membrane transporters on the basis of the transporter classification (TC) scheme, and the Swiss Protein (Swiss-Prot) [ABW+04] database do not contain the same transporters. As demonstrated in Figure 4, less than 40% of the entries in the TCDB are in the Swiss-Prot database. Of those, only 50% are annotated with the Gene Ontology (GO) molecular function (MF) *transporter activity* annotation. Likewise, entries with the same GO annotation in the Swiss-Prot database do not necessarily belong to the same TCDB family. Such inconsistencies complicate the identification of transporters and the prediction of their functions.

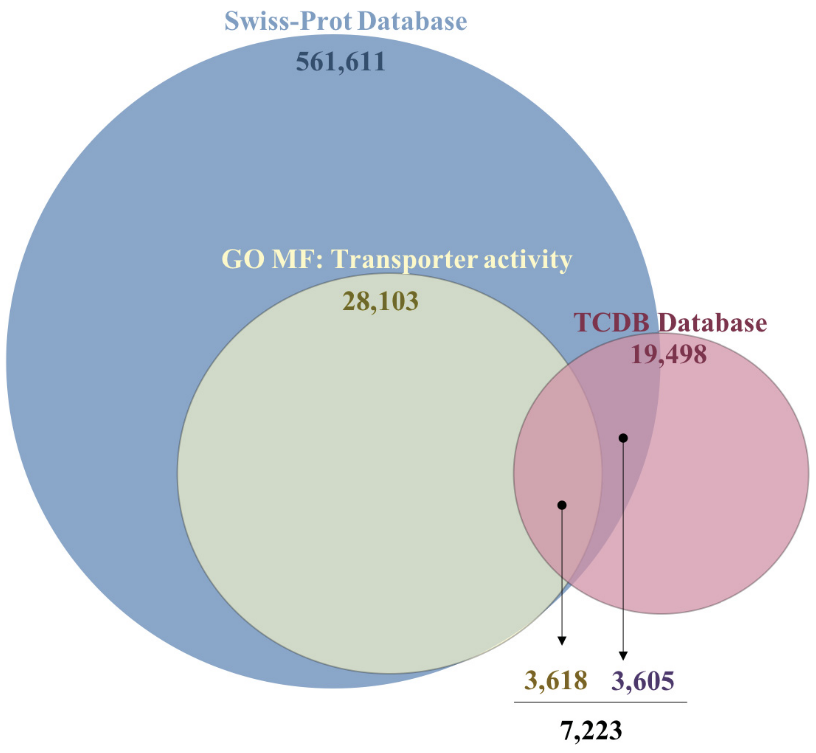


Figure 4: Inconsistencies among the gold standard databases
This figure highlights the inconsistencies among the annotations in the gold standard transporter databases `TCDB` and `Swiss-Prot`. As of February 2020, the `TCDB` contains 19,498 entries, but fewer than 40% of these (7,223 entries) are in the `Swiss-Prot` database. Of those, only 50% (3,618 entries) are annotated with the Gene Ontology (`GO`) molecular function (MF) *transporter activity* annotation.

Generally, there are two methods for predicting transporters: (1) those based on the TC family and (2) those based on the substrate transported across the membrane. The prediction based on the TC family attributes a given protein to a TC functional family. 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.

Predicting the function of a given transporter and getting to the level of substrate specificity for a transporter is challenging, as it is dependent on a very small number of sites in the protein sequence, and those sites are not known beforehand. For example, the crystal structure and the transport mechanism of *Escherichia coli*'s uracil transporter **UraA**, (UniProt-ID `P0AGM7`) was published in 2011 [LLJ+11]. It contains 14 TMSs and is 429 amino acids long. A short pair of antiparallel β-strands, located in TMS3 and TMS10, provide a shelter for substrate binding and have an important role in structural organization and substrate recognition. The uracil binding sites are identified at positions 73, 241, 289, and 290 of the amino acid string, as illustrated in Figure 5. These binding sites are far from each other in the primary sequence but close to each other in the 3D structure of the proteins; thus, identifying them based merely on the primary structure is extremely challenging. This explains why traditional sequence similarity methods do not perform well in identifying the substrate specificity of a transporter [MCCD13] [BH13] [COLG11].

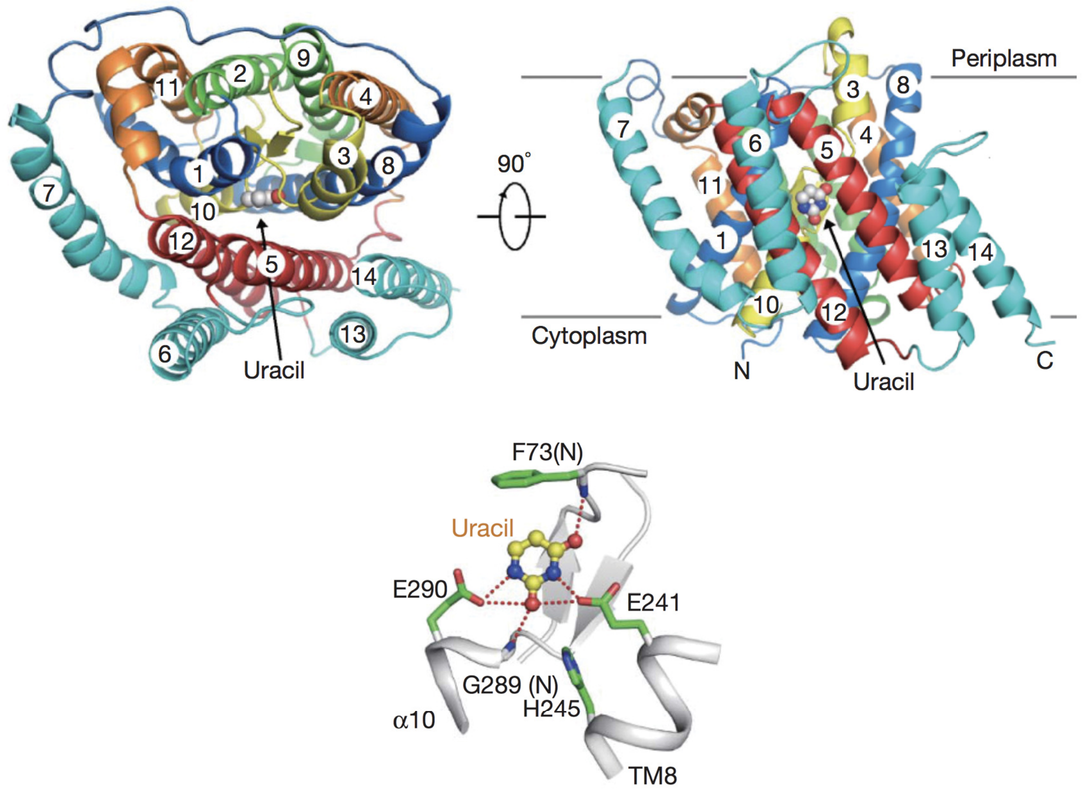


Figure 5: Uracil binding sites on UraA

Top: Overall structure of UraA with 14 TMSs; two perpendicular views, one from the periplasm and one from the side. Bottom: Uracil is coordinated by polar contacts. The uracil molecule is shown in yellow in ball-and-stick form. In addition, two Glu residues, Glu 241 and Glu 290, anchor uracil by each making two hydrogen bonds with it. Replacement of either Glu residue by Ala completely abrogated uracil binding [...]. In addition, the two oxygen atoms of uracil form hydrogen bonds with the amide nitrogen atoms of Phe 73 and Gly 289." [LLJ+11]. This figure is reprinted by permission from Springer Nature [LLJ+11].

This thesis focuses on the *de novo* prediction of the substrate specificity of a transporter. The *de novo* prediction provides insights into the function of novel *unannotated* proteins. Substrate specificity is essential for determining the role of a transporter, as it provides important information on the annotation of proteins and is a key element in transport reactions during network modeling. In the research on identifying the substrate specificity of transporters, there is no universally defined standard dataset. Researchers use their own defined subset of substrate classes, where the assignment of a substrate to substrate classes is not explained or defined, making the actual problems addressed by each predictor diverse, and thus, the expansion of datasets to encapsulate more substrates is almost impossible. To this end, one of the major objectives of this research is to standardize the collection of transporter substrate data so that it is traceable and reproducible. In addition, there are three main questions to be answered regarding a protein of interest:

Q1: Given protein sequence X, is it a membrane protein?

Q2: Given membrane protein sequence X, is it a transporter protein?

Q3: Given transporter protein sequence X, what type of substrates does it transport across the membrane?

Each question needs to be addressed independently because methods for answering one question may not be optimal for the other questions. Therefore, we established a three-layer structure; each layer offers a tailored solution to each question. This design allows for assigning a substrate to a query protein without any prior knowledge, enables obtaining information at the level of interest, and provides the flexibility to skip levels if certain information about the query protein is already known.

4 Objectives

The objectives of this thesis are summarized below:

O1: To improve the computational approaches for detecting *de novo* membrane proteins, relying only on the protein primary sequence.

O2: To improve the computational approaches for detecting *de novo* transporter proteins, relying only on the protein primary sequence.

O3: To facilitate the data collection process for the substrate specificity of transporters in a traceable and reproducible manner.

O4: To broaden the scope of the state-of-the-art for substrate class prediction while maintaining credible predictive performance.

5 Outline

This thesis is organized as follows:

Chapter 2 summarizes the related literature. Section 1 examines the state-of-the-art tools in predicting membrane proteins: Section 1.1 presents the tools for transmembrane topology prediction, and Section 1.2 describes the tools for membrane structural type prediction. Section 2 reviews the efforts to detect transporter proteins, and Section 3 reviews the efforts to predict the substrate specificity of transporters.

Chapter 3 presents important background for this research. Section 1 describes common ways to encode a protein sequence into a numerical vector. Section 2 and Section 3 list important protein databases and ontologies, respectively. Section 4 explains the substitution matrices central to protein comparisons. Section 5 describes the similarity search using the Basic Local Alignment Search Tool (BLAST). Finally, Section 6 presents different multiple sequence alignment (MSA) algorithms.

Chapter 4 explores the best techniques for predicting membrane proteins. Section 1 offers an introduction to the problem and points out the main contributions we make in the chapter. Section 2 introduces a new membrane dataset (*DS-M*) and lists the materials and methods utilized in the experiments. Section 3 delineates the experimental design, and Section 4 presents the results and reveals that an integrative approach, which we call *TooT-M*, outperforms all of the other methods. Section 4.5 compares the results obtained by *TooT-M* with those obtained by the state-of-the-art membrane predictors. The contents of this chapter have been submitted for publication in *BioMed Central (BMC) Bioinformatics.*

Chapter 5 focuses on distinguishing transporter proteins from other membrane proteins. Section 1 provides an introduction to the work and highlights the main contributions we make in the chapter. Section 2 provides an overview of the proposed tool

(*TooT-T*). Section 3 lists the materials and methods utilized in *TooT-T* and introduces a new method of encoding a protein sequence, *psi-composition*, which combines the traditional compositions with evolutionary information obtained by a PSI-BLAST search. Section 4 presents and discusses the results, and Section 4.4 compares the results attained by *TooT-T* with those obtained using the state-of-the-art tools. Finally, Section 5 concludes the chapter. The main components of this chapter have been published in *BMC Bioinformatics* [AB20]. The main difference is that the experiments in the accepted manuscript were performed using the TrSSP benchmark dataset, while in this chapter, the experiments were conducted with a newly constructed benchmark dataset that has the same membrane data as those in *DS-M* from Chapter 4. The results are consistent across both datasets.

Chapter 6 addresses the data collection process for substrate-specific transport proteins. Section 1 introduces the chapter. Section 2 delineates the challenges faced when building a substrate-specific transport protein dataset and highlights the inconsistencies among the gold standard databases. Section 3 elucidates the proposed ontology-based tool, *Ontoclass*. Section 4 presents two case studies; the first case study (Section 4.1) compares *Ontoclass* annotation with a manually curated dataset, and the second case study (Section 4.2) reflects the number of annotated transporters and their substrates in the `Swiss-Prot` database. Section 5 discusses the findings. The contents of this chapter were presented at the 2019 *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* [AB19].

Chapter 7 describes transporter substrate specificity prediction. Section 1 provides an introduction to transporter substrate prediction. Section 2 describes the materials and methods utilized to build the proposed tool, *TooT-SC*. Section 3 presents and analyzes the results, and Section 3.2 compares the performance of *TooT-SC* with the state-of-the-art methods. Finally, Section 4 concludes the chapter. Some contents of the chapter have been published in *PLoS ONE* [AAB20]. The main difference is that the published article proposes *TranCEP*, which utilizes the same approach as *TooT-SC* (TMC-TCS-PAAC) but is trained using the TrSSP dataset to predict <u>seven</u> substrate classes. By contrast, *TooT-SC* is trained using a new dataset, *DS-SC*, with <u>eleven</u> substrate classes. *DS-SC* contains the transporter entries from Chapter 5 annotated using *Ontoclass*.

Finally, **Chapter 8** concludes the thesis, highlights the main contributions, and provides insights into future directions.

Chapter 2

Literature review

This chapter reviews the literature related to this research. Section 1 reviews the efforts to predict membrane proteins. Section 2 reviews the efforts to detect transporter proteins, and Section 3 reviews the efforts to predict the substrate specificity of transporters.

1 Identifying membrane proteins

1.1 Transmembrane topology prediction

Transmembrane topology prediction methods predict the number of TMSs and their respective positions in the primary protein sequence. Transmembrane proteins are IMPs that span the lipid bilayer and have exposed portions on both sides of the membrane. It is expected that the portions that span the membrane contain hydrophobic (nonpolar) amino acids, while the portions that are on either side of the membrane consist mostly of hydrophilic (polar) amino acids. The TMSs can have either α-helical or β-barrel structures, so prediction methods are classified into α-helix prediction methods and β-barrel prediction methods.

Previous prediction methods depended solely on simple measurements such as the hydrophobicity of the amino acids [KD82]. Major improvements were made after the "positive-inside rule" [vH92] was introduced by Von Heijne, which came from the observation that positively charged amino acids, such as arginine and lysine, tend to appear on the cytoplasmic side of the lipid bilayer. Current methods combine hydrophobicity analysis and the positive-inside rule together with machine learning techniques and evolutionary information.

For example, the membrane protein structure and topology support vector machine MEMSATSVM method [NJ09], introduced in 2009, uses four support vector machines (SVMs) to predict transmembrane helices, inside and outside loops, re-entrant helices and signal peptides. In addition, it includes evolutionary information on many homologous protein sequences in the form of a sequence profile. This method outputs predicted topologies ranked by the overall likelihood and incorporates signal peptide and re-entrant helix prediction. The reported accuracy is 89% for the correct topology and location of TM helices and 95% for correct number of TM helices. However, recent studies using experimental data report that MEMSATSVM does not perform as well when evaluated on different datasets [THKE12] [TPS+15].

State-of-the-art methods use consensus algorithms that combine the outputs from different predictors. The consensus prediction of membrane protein topology (TOPCONS2) method [TPS+15] achieved the highest reported prediction accuracy based on benchmark datasets [TGB+18]. It successfully distinguishes between globular and transmembrane proteins and between transmembrane regions and signal peptides. In addition, it is highly efficient, making it ideal for proteome-wide analyses. The TOPCONS2 method combines the outputs from different predictors that can also predict signal peptides (namely, Philius [RKR+08], PolyPhobius [KKS05], OCTOPUS [VE08], signal peptide OCTOPUS (SPOCTOPUS) [VBSE08], and SCAMPI [BVF+08]) into a topology profile where each residue is represented by one of four values: the signal peptide (S), a membrane region (M), the inside membrane (I), or outside membrane (O). Then, a hidden Markov model is used to process the resulting profile and predict the final topology with the highest-scoring state path.

Regarding β-barrel membrane protein prediction, a variety of methods have been introduced, such as methods that combine statistical propensities [BFJE04], k-nearest neighbor (KNN) methods [HY08], neural networks [JMF+01] [OGCS08], hidden Markov models [BLSH04] [SGW+11] [HE12] [TEB16], SVMs [OCG10], and amino acid compositions (AACs) [GAW05] [Lin08]. Approaches based on hidden Markov models have been found to achieve statistically significant performance when compared to other types of machine learning techniques [BLH05]. Major methods for detecting β-barrel outer membrane proteins are HHomp [RLLS09], β-barrel protein OCTOPUS (BOCTOPUS) [HE12], and PRED-TMBB2 [TEB16], with reported MCCs of 0.98, 0.93, and 0.92, respectively, when applied to the same dataset. The BOCTOPUS and HHomp techniques are much slower than PRED-TMBB2 [TEB16].

1.2 Prediction of the membrane protein structural type

Methods for predicting membrane type can predict up to eight different membrane protein structural subtypes categorized as single-pass types I, II, III, and IV; multipass transmembrane; glycophosphatidylinositol (GPI)-anchored; lipid-anchored; and peripheral membrane proteins (Figure 6). A comprehensive review by Butt et al. [BRK17] elucidates these methods in detail. Generally, prediction is performed in two stages: the first stage identifies the protein sequence as membrane or nonmembrane, while the second stage differentiates among specific membrane protein subtypes. This research focuses on detecting all membrane proteins, regardless of their type (the first stage). The state-of-the-art predictors that have achieved the highest overall performance are MemType-2L [CS07] and iMem-2LSAAC [AHJ18].

The MemType-2L [CS07] predictor was introduced in 2007 by Chou and Shen. It is a two-layer predictor that uses the first layer to identify a query protein as a membrane or nonmembrane protein. Then, if the protein is predicted as a membrane protein, the second layer identifies the structural type from among the eight categories. The MemType-2L predictor incorporates evolutionary information by representing the protein samples with pseudo position-specific scoring matrix (Pse-PSSM) vectors and combining the results obtained by individual optimized evidence-theoretic KNN (OET-KNN) classifiers. It achieved an overall accuracy of 92.7% in the membrane detection layer. The reported performance in the first layer is obtained by applying the jackknife test on the provided dataset.

Butt et al. [BKJ$^+$16] introduced a tool that predicts all types of membrane proteins; it uses statistical moments to extract features from the protein samples and then trains a multilayer neural network with backpropagation to predict the membrane proteins. This tool achieved an overall accuracy of 91.23% when applying the jackknife test on the dataset from Chou and Shen [CS07], which was a slightly lower performance than the MemType-2L predictor.

The iMem-2LSAAC was introduced in 2017 by Arif et al. [AHJ18]. iMem-2LSAAC is a two-layer predictor that uses the first layer to predict whether a query protein is a membrane protein. Then, in the case of membrane proteins, it continues to the second layer to identify the structural category. It utilizes the split amino acid composition (SAAC) to extract the features from the protein samples and then applies an SVM to train the predictor. iMem-2LSAAC achieved an overall accuracy of 94.61% in the first layer when applying the jackknife estimator on their dataset.

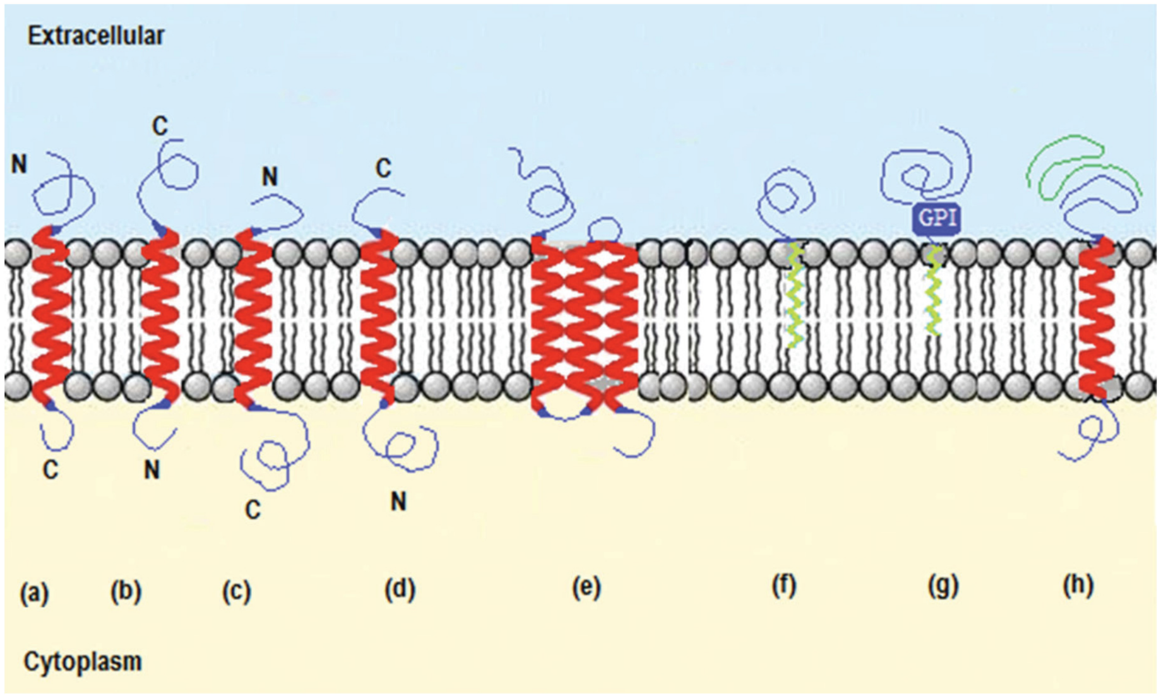


Figure 6: Eight different membrane types

This figure shows the eight types of membrane proteins: (a) single-pass type I: spanning the membrane once, with its N-terminus on the extracellular side of the membrane and its signal sequence removed, (b) single-pass type II: spanning the membrane once, with its N-terminus on the cytoplasmic side of the membrane. The transmembrane domain is located close to the N-terminus, and it functions as an anchor, (c) single-pass type III: spanning the membrane once, with its N-terminus on the extracellular side of the membrane and no signal sequence, (d) single-pass type IV: spanning the membrane once, with its N-terminus on the cytoplasmic side of the membrane. The transmembrane domain is located close to the C-terminus, and it functions as an anchor, (e) multipass: spanning the membrane more than once, (f) lipid-anchored: bound to the lipid bilayer of a membrane through a posttranslational modification by the attachment of at least one lipid or fatty acid, (g) GPI-anchored: bound to the lipid bilayer of a membrane through a GPI-anchor (glycosylphosphatidylinositol anchor), a complex oligoglycan linked to a phosphatidylinositol group, resulting in the attachment of the C-terminus of the protein to the membrane, and (h) peripheral membrane proteins: physically associated with a membrane via interactions with lipid headgroups at the membrane surface or with another membrane protein. This figure is from [BRK17]; definitions are from UniProtKB subcellular location ontology.

2 Identifying transporter proteins

Earlier efforts in transporter detection applied homology searches of experimentally characterized databases to detect novel transporters, and homology searches are still commonly used by many tools. For example, transporters via annotation transfer by homology (TransATH) [AB17] is a system that automates Saier's protocol [YSJ12,GNY$^+$13, PVL$^+$14] based on sequence similarities. TransATH includes the computation of subcellular localization and improves the computation of TMSs. The parameters of TransATH are chosen for optimal performance based on a gold standard set of transporters and non-transporters from *Saccharomyces cerevisiae*. TransATH achieved an overall accuracy of 71.0%. In addition, Barghash et al. [BH13] annotated transporters at the family and substrate levels from three organisms using sequence similarity and sequence motifs.

A major limitation of homology-based methods, however, is that they can generate false assignments because homologous sequences do not always have significant sequence similarities. Likewise, proteins with high sequence similarities do not always share the same function [WL03]. More advanced methods attempt to overcome the limitations of homology-based methods by utilizing features from the protein sequences that better reflect the relation between the sequences and the target function. For example, TrSSP [MCZ14] is a web server for predicting membrane transport proteins and their substrate category. The TrSSP tool applies an SVM in combination with the amino acid index (AAindex) and a position-specific scoring matrix (PSSM) to predict top-level transporters. It achieved transporter prediction accuracies of 78.99% and 80.00% and MCCs of 0.58 and 0.57 during cross-validation and independent testing, respectively.

The scoring card method (SCM) for membrane transport proteins (SCMMTP) [LVY$^+$15] tool uses a novel SCM that utilizes dipeptide compositions to identify putative membrane transport proteins. The SCMMTP method first builds an initial matrix of 400 dipeptides and uses the difference between positive and negative compositions as the initial dipeptide scoring matrix. This matrix is then optimized using a genetic algorithm. The SCMMTP tool achieved overall accuracies of 81.12% and 76.11% and MCCs of 0.62 and 0.47 in cross-validation and independent testing, respectively.

Li et al. [LLX+16] trained an SVM to predict substrate classes of transmembrane transport proteins by integrating features from PSSM, AAC, biochemical, and GO terms. They achieved an overall accuracy of 98.33% and an MCC of 0.97 on an independent dataset. Their method incorporates GO annotation as a feature, which is likely missing in unannotated sequences.

Ho et al. [HPO+19] applied a word-embedding approach from the field of natural language processing (NLP) to the protein sequences of transporters. The protein sequences are defined using both the word embeddings and the frequencies of biological words. They report an outstanding performance in substrate specificity detection for transporters but not in transporter detection. The accuracy for transporter detection reached only 83.94% during cross-validation and 85.00% with independent datasets.

3 Predicting the substrate specificity of transporters

The studies that classify transporter proteins according to the substrates they transport are quite limited. Additionally, for most tools, there is no available software or source code; therefore, it is difficult to compare the results of different tools.

Schaadt et al. [SCH10] used AAC, pair AAC (PAAC), and pseudo-AAC methods (PseAAC), in addition to amino acid conservation with homologous sequences, called MSA-AAC, to detect different substrate specificities. The MSA-AAC method uses a full MSA of each protein in the dataset built by ClustalW. For this step, a BLAST search on the nonredundant database `nr` was conducted to retrieve homologous sequences; then, sequences with an identity below 25% were removed. The occurrence of every amino acid in all sequences of the alignment was normalized to the number of included amino acids, and a resulting vector of size 20 was considered. The investigations were performed on *Arabidopsis thaliana* transmembrane proteins, and they considered four different substrate classes, amino acids, oligopeptides, phosphates and hexoses, with a total of 61 transporters in the positive dataset. This method relies on the Euclidean distance between the query protein sequence composition and the mean composition of protein sequences of each substrate class to compute a score for each query sequence against each substrate class.

Their approach reached an accuracy of approximately 90%, compared to 60% on randomized data. Although the performance is promising, the dataset used by Schaadt et al. contains limited transporters of only one organism.

Chen et al. [COLG11] utilized AAC, PAAC, and biochemical properties using the AAindex database [KPP$^+$07] along with some evolutionary information in the form of PSSM to classify a transporter into four substrate classes: electrons, proteins/mRNAs, ions and others. Their dataset is not tailored to a specific organism and contains a total of 651 transporters. A neural network was employed to construct the classifier, which achieved an accuracy of approximately 80%.

Schaadt et al. [SH12] found that separating TMSs and non-TMSs when calculating AACs yields an improved accuracy of 80% compared to 76% when the composition is computed for the whole sequence. This method also used *Arabidopsis thaliana* transmembrane proteins and considered the same four substrate classes, amino acids, oligopeptides, phosphates and hexoses, with a total of 61 transporters.

Barghash et al. [BH13] applied three different approaches: BLAST [AMS$^+$97], which generates alignments that optimize a measure of local similarity; HMMER [FCE11], which searches sequence databases for sequence homologs using probabilistic methods; and MEME [BE$^+$94], which discovers motifs in protein sequences using expectation maximization. These methods, under different thresholds, were used to evaluate whether annotations of transporter substrates could be transferred from one organism to the other. Four substrate classes were considered: metal ions, phosphate, sugar, and amino acid transporters from *Escherichia coli* (72 transporters), *Saccharomyces cerevisiae* (79 transporters), and *Arabidopsis thaliana* (95 transporters). They found that in the use of these methods, sequences tended to match those from their TC families rather than sequences in the same substrate family. Their reported performance was low for substrate-level classification with an F-measure of approximately 40-75%.

Mishra et al. [MCZ14] developed a web server, TrSSP, for predicting the substrate specificity of transporters. Protein sequence features such as AAC, PAAC, the physicochemical composition, the biochemical composition, and PSSM were used to predict the substrate specificity of seven transporter classes: amino acids, anions, cations, electrons, proteins/mRNAs, sugars, and other transporters. A set of 49 selected physical, chemical, energetic, and conformational properties were used to define the biochemical composition of each protein sequence. The 49 values were selected from the AAindex database [KK00] and have been successfully applied in many areas of bioinformatics, such as protein folding and transporter classification [COLG11]. The normalized values for the 49 amino acid properties were used. The SVM model that was trained using a combination of the biochemical composition and PSSM achieved the highest performance, with an average MCC of 0.41.

Li et al. [LLX+16] used an SVM to predict substrate classes of transmembrane transport proteins by integrating features from PSSM, AAC, biochemical properties, and GO terms. They achieved an overall accuracy of 80.00% on the testing set of the dataset from Mishra et al. [MCZ14]. Their method incorporates GO annotation as a feature, which is likely missing in unannotated sequences.

Ho et al. [HPO+19] applied a word-embedding technique to represent protein sequences of transporters. Their work was the first to use word-embedding representations to classify transporters according to their substrate specificity; however, the concept of using word embeddings to represent a protein sequence has been applied in many studies, such as in [AM15] and [YWBA18]. The idea is motivated by recent advances in natural language processing, where word embeddings, or word vectors, efficiently represent words as low-dimensional floating point vectors in a way that captures their meaning [MCCD13]. In the word-embedding representation of a protein, the amino acid sequence is treated as a sentence of length n, where n is the number of fixed-length words (k-mers) in that sequence. The k-mer words are derived from dividing the sequence into overlapping or nonoverlapping words of length k. Ou et al. divided the protein sequences into overlapping words with a length of 3 (3-mers). Then, the fastText skip-gram model [BGJM17] was trained to learn the individual word vectors for which the dimensionality of the vector was set to 1. The protein sample was then represented by a z fixed-length vector, where z is the number of unique biological words that appear in the corpus. The i^{th} element of this vector represents the 1D word vector representation of the i^{th} word in the corpus multiplied by its frequency in the protein sample. They report an impressive substrate specificity accuracy of 95.25% for an SVM model trained on the TrSSP [MCZ14] training set and tested on its testing set.

Chapter 3

Background

This chapter presents the bioinformatics background related to this research. Section 1 describes common ways to encode a protein sequence into a numerical vector. Section 2 and Section 3 list important protein databases and ontologies, respectively. Section 4 explains substitution matrices that are central to protein comparison. Section 5 describes the similarity search using BLAST. Finally, Section 6 presents different MSA algorithms.

1 Protein composition

Machine learning models generally require their input to be vectors. The conversion from a protein sequence into a numerical vector that encapsulates the protein function is one of the greatest challenges in computational biology. In this section, we describe the baseline features/encodings that are commonly applied in the literature and have been shown to be useful.

The idea of classifying proteins using their AAC was first introduced in 1983 by Nishikawa et al. [NKT83], who found that there is a significant correlation between a protein's AAC and its location, such as inside the cell or outside the cell, as well as its functional properties, such as whether the protein is an enzyme or not. Since then, AACs and their different variations have been used to classify proteins according to many different properties, such as protein structure [NNT86] [TS98] [CZ95], subcellular localization [CAPPQ97], whether a transmembrane protein acts as a channel/pore, electrochemical potential-driven transporters, or primary active transporters [GY08]. Formal definitions of different variations are presented below.

1.1 Amino acid composition (AAC)

The AAC is the normalized occurrence frequency of each amino acid. The fractions of all 20 natural amino acids are calculated by:

$$c_i = \frac{F_i}{L} \qquad i = (1, 2, 3, ...20) \tag{1}$$

where F_i is the frequency of the i^{th} amino acid and L is the length of the sequence. Each protein's AAC is represented as a vector of size 20 as follows:

$$AAC(P) = [c_1, c_2, c_3, ..., c_{20}] \tag{2}$$

where c_i is the composition of the i^{th} amino acid.

1.2 Pair amino acid composition (PAAC)

The PAAC has an advantage over the AAC because it encapsulates information about the fraction of the amino acids as well as their order. It is used to quantify the preference of amino acid residue pairs in a sequence. The PAAC is calculated by:

$$d_{i,j} = \frac{F_{i,j}}{L - 1} \qquad i, j = (1, 2, 3, ...20) \tag{3}$$

where $F_{i,j}$ is the frequency of the i^{th} and j^{th} amino acids of a pair (dipeptide) and L is the length of the sequence. Similar to the AAC, the PAAC is represented as a vector of size 400 as follows:

$$PAAC(P) = [d_{1,1}, d_{1,2}, d_{1,3}, ..., d_{20,20}] \tag{4}$$

where $d_{i,j}$ is the dipeptide composition of the i^{th} and j^{th} amino acids.

1.3 Pseudo-amino acid composition (PseAAC)

The PseAAC was proposed in 2001 by Chou [Cho01] and showed a remarkable improvement in the prediction quality when compared to the conventional AAC. PseAAC is a combination of the 20 components of the conventional AAC and a set of sequence-order correlation factors that incorporate some biochemical properties. Given a protein sequence of length L,

$$R_1R_2R_3R_4...R_L \tag{5}$$

a set of descriptors called sequence-order-correlated factors are defined as follows:

$$\begin{cases} \theta_1 = \dfrac{1}{L-1}\displaystyle\sum_{i=1}^{L-1}\Theta(R_i, R_{i+1}) \\ \theta_2 = \dfrac{1}{L-2}\displaystyle\sum_{i=1}^{L-2}\Theta(R_i, R_{i+2}) \\ \theta_3 = \dfrac{1}{L-3}\displaystyle\sum_{i=1}^{L-3}\Theta(R_i, R_{i+3}) \\ \quad\quad . \\ \quad\quad . \\ \quad\quad . \\ \theta_\lambda = \dfrac{1}{L-\lambda}\displaystyle\sum_{i=1}^{L-\lambda}\Theta(R_i, R_{i+\lambda}) \end{cases} \tag{6}$$

The parameter λ is chosen such that $(\lambda < L)$. The correlation function is given by:

$$\begin{aligned}\Theta(R_i, R_j) = \frac{1}{3}\{&[H_1(R_j) - H_1(R_i)]^2 + [H_2(R_j) - H_2(R_i)]^2 \\ &+[M(R_j) - M(R_i)]^2\}\end{aligned} \tag{7}$$

where $H_1(R_i)$ is the hydrophobicity value, $H_2(R_i)$ is the hydrophilicity value, and $M(R_i)$ is the side-chain mass of the amino acid R_i. These quantities were converted from the original hydrophobicity value, the original hydrophilicity value and the original side-chain mass by a standard conversion formula as follows:

$$H_1(R_i) = \frac{H_1^\circ(R_i) - \frac{1}{20}\sum_{k=1}^{20} H_1^\circ(R_k)}{\sqrt{\frac{\sum_{y=1}^{20}\left[H_1^\circ(R_y) - \frac{1}{20}\sum_{k=1}^{20} H_1^\circ(R_k)\right]^2}{20}}} \tag{8}$$

where $H_1^\circ(R_i)$ is the original hydrophobicity value for amino acid R_i and can be taken from the work of Tanford [Tan62]; $H_2^\circ(R_i)$ and $M^\circ(R_i)$ are converted to $H_2(R_i)$ and $M(R_i)$, respectively, in the same way. The original hydrophilicity value $H_2^\circ(R_i)$ for amino acid R_i can be obtained from Hopp and Woods [HW81]. The mass $M^\circ(R_i)$ of the R_i amino acid side chain can be obtained from any biochemistry textbook. PseAAC is represented as a vector of size $(20 + \lambda)$ as follows:

$$PseAAC(P) = [s_1, ..., s_{20}, s_{21}, ..., s_{20+\lambda}] \tag{9}$$

where s_i is the pseudo-AAC as follows:

$$s_i = \begin{cases} \dfrac{f_i}{\sum_{r=1}^{20} f_r + \omega \sum_{j=1}^{\lambda} \theta_j} & 1 \leq i \leq 20 \\ \\ \dfrac{\omega\theta_{i-20}}{\sum_{r=1}^{20} f_r + \omega \sum_{j=1}^{\lambda} \theta_j} & 20 < i \leq 20 + \lambda \end{cases} \tag{10}$$

where f_i is the normalized occurrence frequency of the *ith* amino acid in the protein sequence, θ_j is the j^{th} sequence-order-correlated factor calculated from Equation 6, and ω is a weight factor for the sequence-order effect. The weight factor ω puts weight on the additional PseAAC components with respect to the conventional AAC components. The user can select any value from 0.05 to 0.7 for the weight factor. The default value given by Chou [Cho01] is 0.05.

2 Databases

2.1 Transporter classification database (TCDB)

The TCDB [SJTB06] uses a classification system approved by the International Union of Biochemistry and Molecular Biology (IUBMB) for membrane transport proteins; it is known as the TC system. The TCDB is a curated database of accurate and experimentally characterized information collected from over 10,000 published references. As of March 2020, it contains more than 19,500 unique protein sequences classified into more than 1,448 transporter families. Each entry in the database has a TC identifier (TCID) that consists of five components, V.W.X.Y.Z., where V is a number from 1 to 9 that corresponds to the transporter class (e.g., channels, carrier, pumps (active transport)), W refers to a transporter subclass, X is a number that refers to the transporter family, Y is a number that corresponds to the transporter subfamily, and Z refers to the substrate or range of substrates transported. Figure 7 shows an example TCDB entry. The transporter class

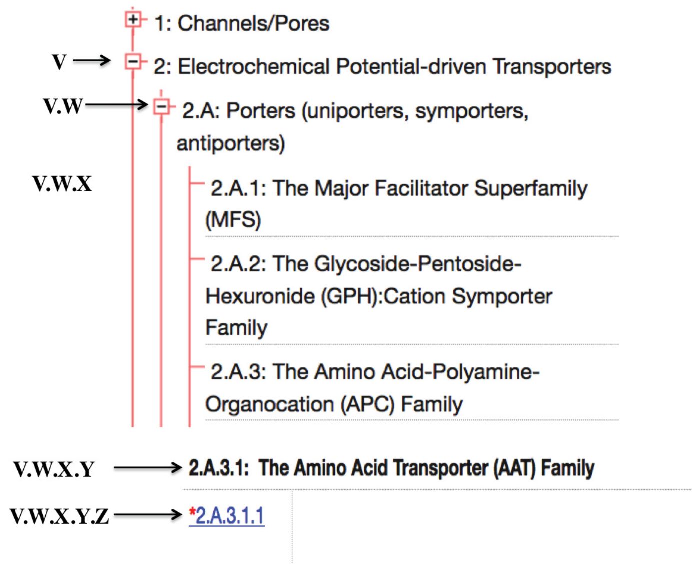

1: Channels/Pores
V
2: Electrochemical Potential-driven Transporters
V.W
2.A: Porters (uniporters, symporters, antiporters)
V.W.X
2.A.1: The Major Facilitator Superfamily (MFS)
2.A.2: The Glycoside-Pentoside-Hexuronide (GPH):Cation Symporter Family
2.A.3: The Amino Acid-Polyamine-Organocation (APC) Family
V.W.X.Y
2.A.3.1: The Amino Acid Transporter (AAT) Family
V.W.X.Y.Z
*2.A.3.1.1

Figure 7: TCDB entry example

The TCID consists of five components: V.W.X.Y.Z. V is a number from 1-9 that corresponds to the transporter class (e.g., channels, carrier, pumps (active transport)), W is a letter that refers to the transporter subclass, X is a number that refers to the transporter family, Y is also a number that corresponds to the transporter subfamily, and Z refers to the substrate or range of substrates.

TC.9 contains all of the uncharacterized transporters and has approximately 3,266 entries.

2.2 Universal Protein Resource Knowledgebase (UniProtKB)

The UniProtKB [ABW+04] is the primary worldwide database of protein sequences and highly annotated functional information. UniProtKB employs GO annotation, which associates GO terms (see Section 3.2) with UniProtKB records. This association is accompanied by the reference from which the evidence is derived, in addition to the evidence code that indicates the degree to which the annotation is supported. The evidence codes are commonly encoded as three-letter "GO evidence codes". However, they are now being replaced by Evidence and Conclusion Ontology (ECO) (see Section 3.3) terms that provide the ability to capture more in-depth evidence information than traditional GO evidence codes. Furthermore, 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. Generally, UniProtKB consists of two sections: Swiss-Prot and TrEMBL.

Swiss-Prot contains well-annotated, nonredundant proteins that have been manually inspected. 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. Most GO annotations in the Swiss-Prot database are supported by ECO manual curation terms.

TrEMBL contains unrevised and automatically annotated protein sequences. All of the GO terms are associated with ECO terms based on automatic assertion. In addition, TrEMBL entries generally contain fewer keywords than Swiss-Prot entries, and the keywords are assigned automatically according to specific annotation rules.

As of March 2020, Swiss-Prot contains 561,611 sequence entries, and TrEMBL contains 177,754,527 sequence entries.

3 Ontologies

3.1 Chemical entities of biological interest (ChEBI)

ChEBI [HOD+15] is a database and ontology that contains information about chemical entities. Each entry in the database is classified within the ontology. ChEBI serves as an annotation source of unique and reliable identifiers for chemicals. It is commonly used in many bioinformatics databases and as a chemistry component of several ontologies, including GO [HAB+13]. The ChEBI ontology contains three subontologies:

- A chemical ontology in which entities are classified based on their structural features and properties (e.g., monosaccharide, carboxylic acid, or anion);
- A role ontology in which the entities are classified based on their activities in chemical or biological systems (e.g., vitamin, drug, or enzyme);
- A subatomic particle ontology in which particles smaller than atoms are classified (e.g., photons or nucleons).

ChEBI uses nomenclature, symbolism, and terminology endorsed by the International Union of Pure and Applied Chemistry (IUPAC) and the Nomenclature Committee of the IUBMB (NC-IUBMB). The primary ontology relationships are *"is a"* relationship, which is for classification, *"has part of"* relationship, which links composite entities to their component parts, and *"has a role"* relationship, which links a chemical entity to a role in the role ontology. Figure 8 displays a graph view of a hexose (CHEBI:18133).

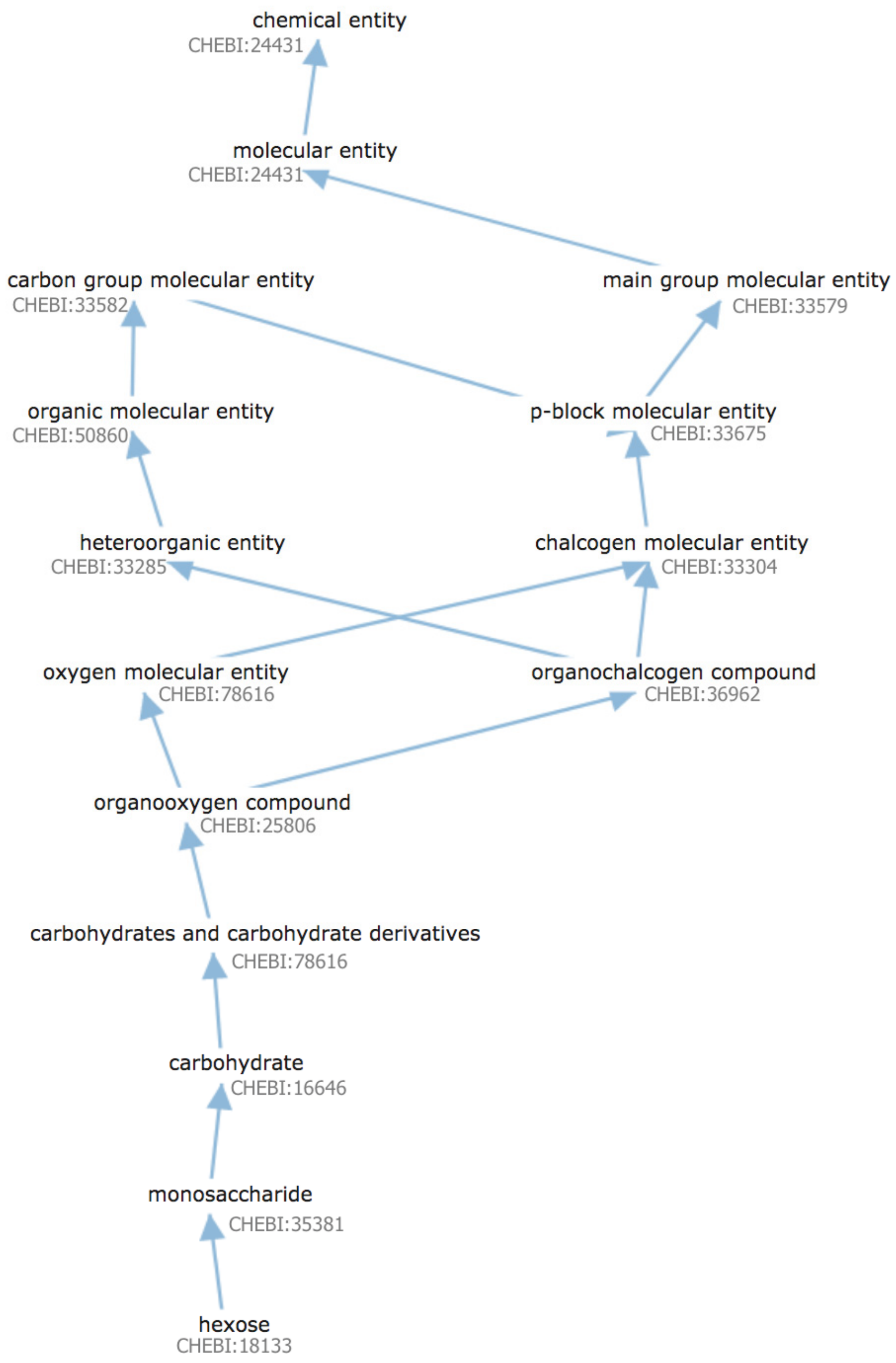


Figure 8: Example of a `ChEBI` ontology graph view.

This figure illustrates a full graph view of a *hexose* (`CHEBI:18133`) in the `ChEBI` ontology. All of the presented relationships in this case are *"is a"* relationships between the ontologies. By clicking on ontology terms within the view, one can see a definition for that specific term.

3.2 Gene Ontology (GO)

The `GO` Project [ABB+00] is the largest resource available that provides an ontology of defined terms representing gene product properties. The `GO` Project describes functions with respect to three domains: molecular function (MF), biological process (BP) and cellular component (CC). The MF concerns the activities of a gene product at the molecular level. The BP term includes the larger processes accomplished by multiple molecular activities. CC encompasses the locations relative to cellular structures in which a gene product performs its function. The ontology is structured as a directed acyclic graph in which each term has a specific relationship to one or more other terms in the same domain. The `GO` terms that refer to chemical entities have fully defined semantic relationships with corresponding chemical terms in `ChEBI`. This correspondence is intended to facilitate an accurate and consistent, system-wide chemical view of the biological representation [HAB+13].

Transporter-related terms include the `GO` MF term `GO:0005215 transporter activity`, which is defined as "the function that enables the directed movement of substances (such as macromolecules, small molecules, ions) into, out of, or within a cell, or between cells" and the `GO` BP term `GO:0006810 transport`.

3.3 Evidence and Conclusion Ontology (ECO)

`ECO` is a structured, controlled vocabulary for capturing evidence in biological research. This ontology is used to document evidence-based conclusions derived from investigations [CMB+14]. The current version of `ECO` contains more than 600 terms arranged in a hierarchy with two high-level classes, namely, *evidence* (ECO:000000) and *assertion method* (ECO:0000217). *Evidence* is defined as a "type of information that is used to support an assertion". The majority of evidence is either experimental (e.g., expression pattern evidence) or computational (e.g., sequence similarity evidence); other types include author statements (with or without traceable reference) and curator inferences. In addition to the evidence, `ECO` describes the mechanism by which an assertion is made (e.g., manually by a curator or electronically). The *assertion method* is defined as "a means by which a statement is made about an entity". For example, if an algorithm was used to assign a predicted function to a protein without any curator judgment, ECO expresses this situation as an automatic assertion. Similarly, if a curator creates the annotation after reading a result reported in a paper, ECO captures that as a manual curation.

4 Substitution matrix

The amino acid similarity can be thought of in terms of chemical similarity [KYB03]. That is, amino acids that share similar chemical properties are more related than those with different properties. Figure 9 delineates a rough qualitative representation of the chemical relationships of amino acids. From an evolutionary point of view, one would expect mutations that completely change the chemical properties of an amino acid to be less common, as they may alter the protein's 3D structure; while changes between similar amino acids happen more frequently [KYB03].

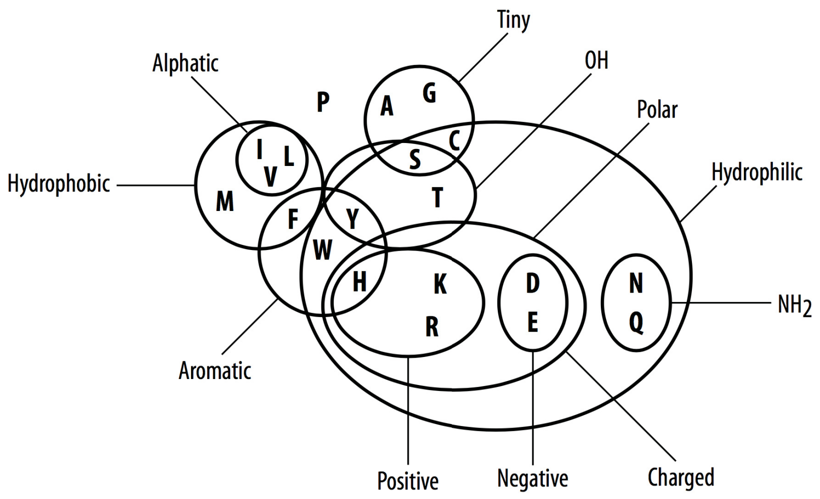
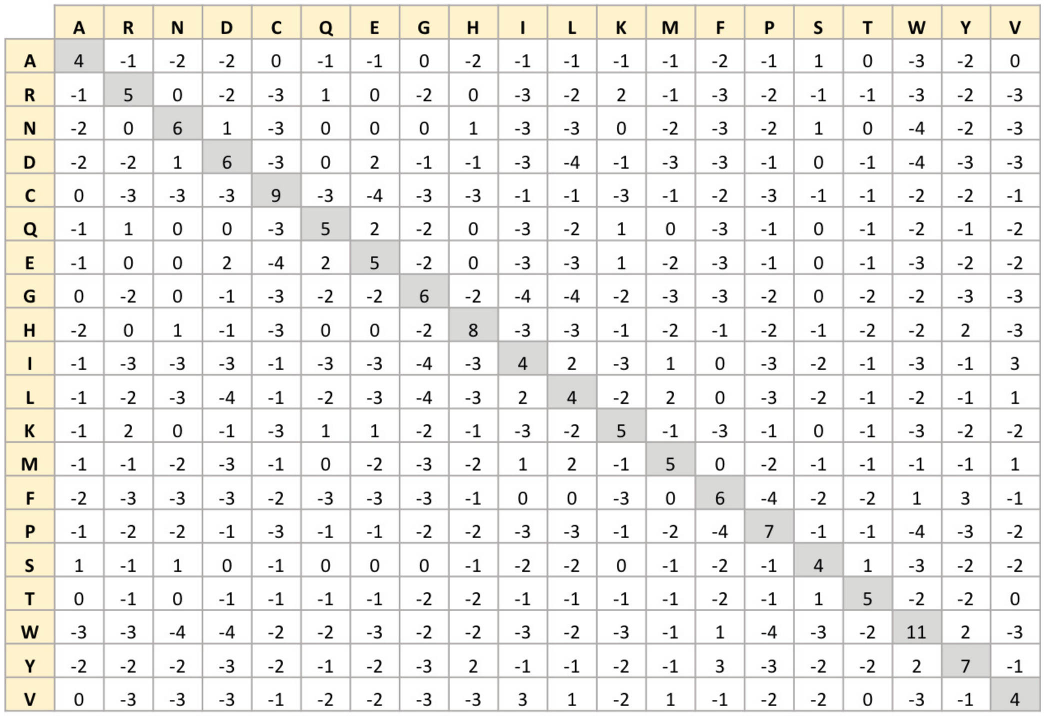


Figure 9: Chemical relationships among amino acids [KYB03]

The scoring matrix contains a quantitative measure of amino acid similarity. Establishing a scoring matrix based on the general chemical properties of amino acids is possible; however, it does not account for substitutions that are more likely to happen from an evolutionary standpoint. Thus, most of the widely used scoring matrices are based on the observed conservation of amino acids over time, such as point accepted mutations (PAM) and blocks substitution matrices (BLOSUMs). Since these matrices represent relative rates of evolutionary substitutions, they are also called substitution matrices. An example of a BLOSUM substitution scoring matrix is shown in Figure 10. The scores in the matrix represent the log odds ratio of the observed probability of a substitution of a pair of amino acids divided by the probability expected purely due to chance, calculated as follows:

$$S_{i,j} = \frac{1}{\lambda} log \frac{q_{i,j}}{p_i p_j} \tag{11}$$

where $q_{i,j}$ is the probability of amino acids i and j replacing each other in a homologous sequence and p_i and p_j are individual probabilities of amino acids i and j, respectively; λ is a scaling factor, set such that the matrix contains easily computable integer values. Substitution matrices are key in protein comparison, identifying homologs, and scoring in



	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Figure 10: BLOSUM62 substitution matrix
BLOSUM62 indicates a BLOSUM computed by choosing blocks that are more than 62% identical.

any sequence alignment algorithm.

5 Basic Local Alignment Search Tool (BLAST)

BLAST [AMS+97] is one of the most popular sequence search programs in bioinformatics. It is used to compare primary sequence information and find regions of local similarity between sequences. BLAST uses a heuristic method to identify homologous sequences by searching for short similarity regions and expanding the hits in both directions.

Given a query protein Q, BLAST first compiles a list of overlapping words of length w and finds neighborhood words whose scores are greater than a threshold t. The default setting for w is 11 for DNA alignment and 3 for protein alignment. Next, the database is scanned for exact matching words with the compiled list of words. The matches are extended in both the left and right directions until the score drops below a predetermined threshold x. The alignment that has a score above the threshold is called *the highest-scoring pair* (HSP). The scores are determined according to a substitution matrix, and the default is BLOSUM62. Finally, BLAST calculates the statistical significance of the HSP as the expected value (e-value) and sorts the results according to it. The e-value describes the likelihood that a given score would occur by chance and is calculated as follows:

$$e\text{-}value = Kmne^{\lambda S} \tag{12}$$

where S is the alignment score of the HSP; m and n are the query length and the length of the database, respectively; λ is a parameter that scales the scoring system; and K is a scaling factor. λ and K are based on the Karlin-Altschul theory, thus, they are often called Karlin-Altschul statistics.

The e-value in Equation 12 decreases exponentially as the alignment score increases. In addition, the relationship between the e-value and the search space (mn) is linear; that is, if the search space is doubled, the e-value is also doubled.

6 Multiple sequence alignment (MSA)

MSAs are fundamental tools for protein structure, function prediction, phylogenetic analysis, and other bioinformatics and molecular evolutionary applications. An MSA is a collection of more than two protein sequences that are partially or completely aligned into a rectangular array. The goal of an MSA is to align the sequences in such a way that the residues in a given column are homologous in an evolutionary sense (driven from the same residue of the shared ancestry), homologous in a structural sense (occupying same positions in the 3D structure) or have a common function. In closely related sequences (40% amino acid identity or more), these three principles are essentially the same. On the other hand, if the protein sequences show some divergence over evolutionary time, those principles may result in considerably different alignments, and the MSA becomes extremely difficult to solve [EB06] [Pev09]. MSA development is an active area of research; over the past decade, dozens of algorithms have been introduced. The most popular MSA algorithms are reviewed here.

The exact methods use dynamic programming to find the global optimal alignment with time complexity, $O(L^N)$, where L is the average sequence length and N is the number of aligned sequences. Since time grows exponentially as N increases, these methods are not feasible for use unless N is very small [KT08].

ClustalW [THG94], one of the most popular MSA heuristic algorithms, uses a progressive method. First, the algorithm performs a pairwise alignment of all the sequences in the alignment in a matrix that shows the similarity of each pair of sequences. The similarity scores are usually converted into distance scores. Second, the algorithm uses the distance score matrix to construct a rough phylogenetic tree called a guide tree. Finally, ClustalW progressively aligns the sequences by following the branching order of the guide tree. Progressive methods are very efficient, and hundreds of sequences can be aligned rapidly with these methods. However, when an error is introduced in the early stages of the alignment, it cannot be corrected, which may increase the likelihood of misalignment due to incorrect conservation signals [Pev09] [DOS13].

Clustal Omega [SWD$^+$11], the latest algorithm from the Clustal family, is highly efficient and more accurate than ClustalW. Clustal Omega is capable of aligning more than 190,000 sequences on a single processor in a few hours [SWD$^+$11]. Like ClustalW, Clustal Omega first performs a pairwise alignment. Then, to reduce the number of distance calculations required to build the guide tree, Clustal Omega uses a modified version of mBed [BSS$^+$10], which involves embedding the sequences in a space where the similarities within a set of sequences can be approximated without the need to compute all the pairwise distances. The sequences can then be clustered extremely quickly to produce the guide tree. Finally, progressive alignments are computed using the HHalign package [Söd05], which aligns the sequences with two hidden Markov model profiles.

Iterative methods overcome the inherited limitation of the progressive method (i.e., the error cannot be removed once introduced). The multiple alignment using fast Fourier transform (MAFFT) algorithm [KMKM02] is an iterative method that uses two-cycle heuristics. Initially, it aligns the sequences using progressive methods and then refines the alignment by calculating and optimizing the sum-of-pairs score. The MAFFT algorithm also identifies homologous regions by a fast Fourier transform, where the amino acid sequence is converted to a sequence with volume and polarity values for each amino acid residue.

The idea behind consistency-based methods is that for sequences x, y and z, if residue x_i aligns with residue y_j and residue y_j aligns with residue z_k, then residue x_i aligns with residue 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. The tree-based consistency objective function for alignment evaluation (T-Coffee) algorithm [NHH00], a consistency-based method, is considered one of the most accurate programs available based on benchmarking studies. T-Coffee takes into account both global and local pairwise alignments. Local similarity is used to reveal when two proteins share part of the sequence, e.g., a domain or motif.

All of the abovementioned algorithms are general-purpose algorithms that can be used to align any related protein sequences. In other words, they use general scoring schemes tailored for sequences of soluble proteins. Because the regions of transmembrane proteins that are inserted into the cell membrane have a profoundly different hydrophobicity pattern compared with soluble proteins, these algorithms may not produce the optimal alignment for transmembrane proteins [PFH08].

Few packages have been published to address the problem of aligning transmembrane proteins, such as PROLIN-TM [PFH08], TM-Coffee [CDTTN12] and the simple transmembrane alignment method (STAM) [SG04]. Most of these algorithms use a homology extension technique. In homology extension methods, database searches are used to replace each sequence with the profile of closely related homologs. Consequently, each sequence position becomes a column in the multiple alignments that reveals the pattern of acceptable mutations. TM-Coffee is the most accurate method based on benchmarking studies performed by Notredame et al. [CDTTN12]. The TM-Coffee algorithm can be summarized as follows: for each sequence in need of alignment, a homology search is performed using BLAST [AMS$^+$97], and the hits with an identity between 50% and 90% and a coverage of more than 70% are retained. Then, the BLAST output is transformed into a profile where all columns corresponding to unaligned positions (i.e., gaps) in the query are removed, and the query positions unmatched by BLAST are filled with gaps. Finally, a T-Coffee library is produced by aligning every pair of profiles. TM-Coffee shows a 10% improvement over MSAProbs [LSM10], which is the next best method that uses homology extension. Although homology extension-based methods achieve much more accurate alignments than standalone methods, performing an alignment takes several orders of magnitude longer [EB06].

The assessment of MSA has been the subject of research in recent years. Particularly, efforts have been devoted to answering two main questions: how to obtain alignments associated with the optimal score and how to evaluate the "goodness" of an alignment. A reliable way to evaluate the alignment is to compare the alignment result with known 3D structures established by X-ray crystallography. Because it has been demonstrated that even proteins with low sequence identity (less than 40%) can share similar 3D structures, a comparison of the 3D structures makes it possible to align distantly related proteins with low sequence similarity on the basis of their structural equivalence [Kri07] [Got96].

Several benchmark datasets have been created as reference sets in which alignments are created from proteins with known structures. In this way, one can evaluate the result of a proposed MSA algorithm on the basis of studied proteins that are experimentally and structurally homologous. Many studies devoted to comparing different MSA algorithms on tests against benchmark databases are currently available [EB06] [TLLP11] [PdROC14]. They can serve as a guide for researchers to choose the appropriate algorithm for a given dataset. The general conclusion is that there is a trade-off between the computational cost and the accuracy; the accuracy can vary greatly if the sequences under study are highly divergent. In addition, there is no available MSA program that outperforms the others in all test cases [PdROC14].

Chapter 4

Identifying membrane proteins

This chapter addresses the first research objective:

O1: To improve the computational approaches for detecting de novo membrane proteins, relying only on the protein primary sequence.

Some of the contents in this chapter were presented at the 2019 Network Tools and Applications in Biology (NETTAB) international conference in Italy and have been submitted for publication in *BMC Bioinformatics*: M. Alballa, G. Butler, Integrative approach for detecting membrane proteins, *BMC Bioinformatics* (under review).

The chapter is organized as follows: Section 1 gives an introduction to the problem and points out the main contributions we make in the chapter. Section 2 introduces a new membrane dataset (*DS-M*) and lists the materials and methods utilized in the experiments, Section 3 delineates the experimental design, Section 4 presents the results and finds that an integrative approach, which we call *TooT-M*, outperforms all the other methods, and Section 4.5 compares the results obtained by *TooT-M* with those obtained by the state-of-the-art membrane predictors. Finally, Section 5 concludes the chapter.

1 Introduction

A major class of membrane proteins are transmembrane proteins (Figure 6 (a)–(e)). These proteins have one or more TMSs embedded in the lipid bilayer in addition to extramembranous hydrophilic segments that extend into the water-soluble domains on each side of the lipid bilayer. The embedded segments are distinguishable because they contain residues with hydrophobic properties that interact with the hydrophobic (nonpolar) tails of the membrane phospholipids. Other classes of membrane proteins include surface-bound proteins that do not extend into the hydrophobic interior of the lipid bilayer; they are typically bound to the lipid head groups at the membrane surface or attach to other IMPs (Figure 6 (f), (g), and (h)). Unlike transmembrane proteins, surface-bound proteins such as peripheral and lipid-anchored proteins do not have TMSs; they are therefore more difficult to distinguish from other globular proteins.

Two distinct approaches, namely, transmembrane topology prediction and membrane structure type prediction, are primarily used to detect membrane proteins. While transmembrane topology tools predict only a subset of membrane proteins (transmembrane proteins), they are applied more often than membrane structure type prediction tools due to the vast number of tools available and because transmembrane proteins constitute a major class of membrane proteins. However, by overlooking other classes of membrane proteins, essential information is lost. By contrast, membrane structure type predictions can be used to detect all classes of membrane proteins. A comprehensive review by Butt et al. [BRK17] discussed these methods in detail. Generally, there are two stages of prediction: the first stage identifies the protein sequence as that of a membrane or nonmembrane protein, while the second stage differentiates specific subtypes of membrane proteins. In this work, we focus on detecting membrane proteins of all types, i.e., the first stage. That is, *given a protein sequence Q, is it a membrane protein?*

The state-of-the-art tools that have achieved the highest overall performance in predicting all types of membrane proteins are MemType-2L [CS07] and iMem-2LSAAC [AHJ18]. While MemType-2L [CS07] has been in use for over a decade, it has maintained its popularity due to its simple yet effective methodology. MemType-2L incorporates evolutionary information by representing protein samples with Pse-PSSM vectors and combining the results obtained from individual OET-KNN classifiers. By contrast, iMem-2LSAAC uses the split AAC to extract features from protein samples and then SVMs to train the predictor.

MemType-2L is the only accessible tool for the prediction of all types of membrane proteins. When we tested it on a new set of membrane proteins, the accuracy reached only 80%. This could be because it was trained on the available protein sequences from 2006; however, the protein sequence landscape has drastically changed. As presented in Figure 11, a large surge in protein sequence entries has been recorded since 2006, and the tool may have missed patterns present in more recent data. It is therefore essential to build a new accessible tool that accommodates all membrane data.

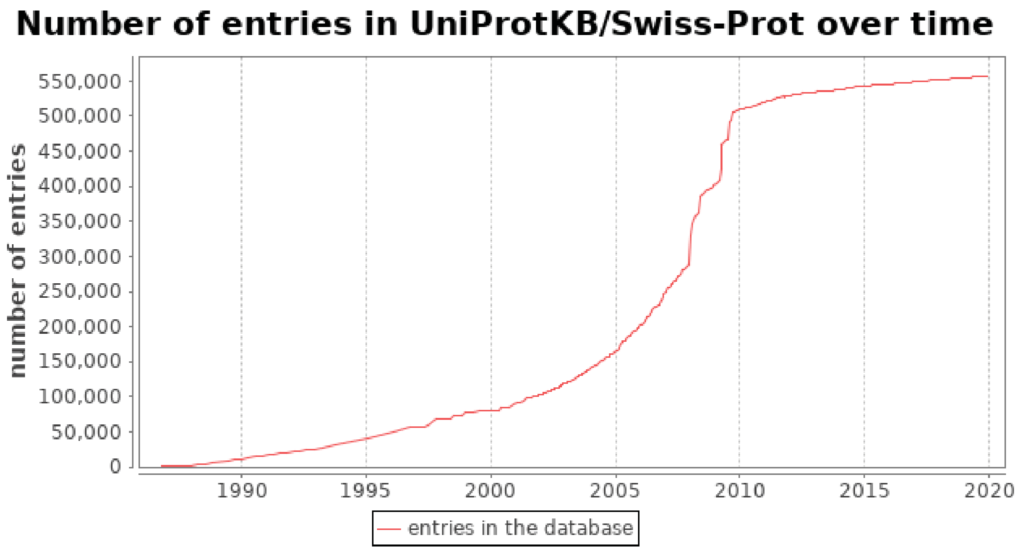


Figure 11: Number of entries in the `Swiss-Prot` database over time. This figure is from the official statistics page on the UniProt website

The main contributions of this work can be summarized as follows:

- We establish a new benchmark dataset for membrane proteins (*DS-M*).
- We evaluate the performances of traditional transmembrane topology prediction tools on *DS-M* to predict all types of membrane proteins.
- We compare the performances of various machine learning techniques to detect membrane proteins; this comparison involved applying different feature extraction techniques to encode protein sequences and choosing the proper machine learning algorithm to build a model on the extracted vectors.
- We introduce a novel method, *TooT-M*, which integrates different techniques that achieves superior performance compared to all other methods, including the state-of-the-art methods.

2 Materials and methods

2.1 Dataset

The latest publicly available benchmark dataset that contains both membrane and nonmembrane proteins was constructed by Chou and Shen [CS07] and was used to construct the MemType-2L predictor. Their dataset was collected from the `Swiss-Prot` database version 51.0, released on October 6, 2006. Furthermore, they eliminated proteins with 80% or more similarity in their sequences to reduce homology bias. Chou and Shen's dataset contains a total of 15,547 proteins, of which 7,582 are membrane proteins and 7,965 are nonmembrane proteins.

Because of the rapidly increasing sizes of biological databases, we built a new updated dataset, *DS-M*. This dataset was collected from the June 2018 release of the `Swiss-Prot` database. The annotated membrane proteins were retrieved by extracting all of the proteins that are located in the membrane, using the following search query:

```
locations:(location:membrane) AND reviewed:yes
```

The remainder of the `Swiss-Prot` entries were designated as nonmembrane proteins. The sequences in both classes were filtered by adhering to the following criteria:

- **Step 1:** Protein sequences that have evidence "inferred from homology" for the existence of a protein were removed.
- **Step 2:** Protein sequences less than 50 amino acids long were removed, as they could be fragments.
- **Step 3:** Protein sequences that have no `GO` MF annotation or annotation based only on computational evidence (inferred from electronic annotation, IEA) were excluded.
- **Step 4:** Protein sequences with more than 60% pairwise sequence identity were removed via a CD-HIT [LG06] program to avoid any homology bias.

All sequences from the membrane class and randomly selected sequences from the nonmembrane class were used to form the benchmark dataset. The data were randomly

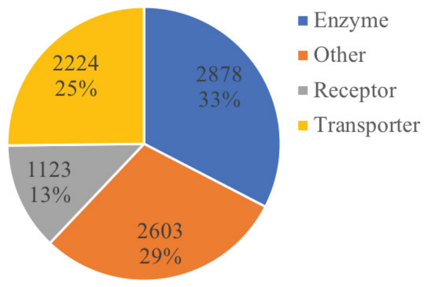


Figure 12: Membrane functional types

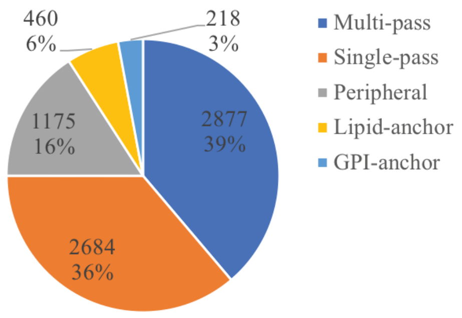


Figure 13: Membrane structural types

divided (stratified by class) into the training (90%) and testing (10%) sets, as illustrated in Table 1. Since the testing set was kept aside until the final testing phase, and was not used during the training or model selection, we refer to it as independent testing set.

Class	**Training**	**Testing**	**Total**
Membrane	7,945	883	8,828
Nonmembrane	8,157	907	9,064
Total	16,102	1,790	17,892

Table 1: Membrane dataset *DS-M*

The dataset contains samples from different species, with the most sequences coming from *Homo sapiens* (18%), *Arabidopsis thaliana* (14%), *Mus musculus* (11%), *Saccharomyces cerevisiae* (8%), and *Saccharomyces pombe* (6%). Enzymes, transporters, receptors, and those annotated with other annotations account for 33%, 25%, 13%, and 29%, respectively, of the membrane data collected, as presented in Figure 12.

Approximately 84% of the membrane data collected have a structural type annotation. Figure 13 indicates that of the annotated proteins, approximately 75% are transmembrane proteins (single or multipass), while the remainder are peripheral, lipid-anchored, or GPI-anchored proteins.

2.2 Topology prediction tools

A protein is regarded as a membrane protein if at least one TMS is detected. With respect to α-helical transmembrane proteins, three tools were applied. TOPCONS2

[TPS$^+$15] is considered the state-of-the-art method and known for its ability to distinguish signal peptides from transmembrane regions. TOPCONS2 results were obtained through its available web server. The second tool is HMMTOP [TS01], which is a highly efficient tool commonly used in the literature. HMMTOP results were also obtained through its web server. The third tool is the transmembrane hidden Markov model (TMHMM) [KLvHS01], which is also commonly applied in the literature, and its results were obtained from its web server.

Regarding β-barrel transmembrane proteins, we applied PRED-TMBB2 [TEB16], which shows comparable performance to the state-of-the-art β-barrel predictors but is much more efficient in terms of the runtime [TEB16]. We used the PRED-TMBB2 web server to obtain the results.

2.3 Protein sequence encoding

After establishing the dataset, it is necessary to find the best representation of the protein sequences and use it to train the prediction engine. Generally, there are two options in representing a protein sequence: sequential or discrete representations [CS07]. In sequential representations, a sample protein is represented by its amino acid sequence and then used in a similarity search-based tool such as BLAST [AMS$^+$97], where the top hits give indication about the function of the query protein. A major drawback of relying on the similarity is that it fails when proteins with the same function share a low sequence similarity. In discrete representations, a sample protein is represented by a set of discrete numbers that are usually the result of feature engineering. In this study, we encoded the protein sequences using the AAC, PAAC, and PseAAC baseline compositions. In addition, we applied the Pse-PSSM and SAAC as described below.

2.3.1 Pseudo position-specific scoring matrix (Pse-PSSM)

We adopted the Chou and Shen [CS07] protein-encoding strategy, Pse-PSSM. The Pse-PSSM is built by first performing a Position-Specific Iterative BLAST (PSI-BLAST) [AMS$^+$97] search on a protein sequence $\mathbf{P}$ using the `Swiss-Prot` database (3 iterations, e-value cutoff of 0.001) and retrieving the PSSM profile:

$$P_{PSSM} = \begin{bmatrix} E_{1\to1} & E_{1\to2} & E_{1\to3} & \dots & E_{1\to20} \\ \vdots & \vdots & \vdots & & \vdots \\ E_{i\to1} & E_{i\to2} & E_{i\to3} & \dots & E_{i\to20} \\ \vdots & \vdots & \vdots & & \vdots \\ E_{L\to1} & E_{L\to2} & E_{L\to3} & \dots & E_{L\to20} \end{bmatrix} \tag{13}$$

P_{PSSM} has L rows (a row for each position in protein sequence $\mathbf{P}$) and 20 columns (one for each amino acid). Each element $E_{i\to j}$ represents the score for the substitution of the amino acid in the i^{th} position of the protein sequence to the amino acid of type j in the evolution process. Since the number of columns in the PSSM depends on the length of the protein sequence $\mathbf{P}$, the Pse-PSSM first standardizes the PSSM scores so that they have a mean value of zero over the 20 amino acids and then uses the following uniform size vector to represent protein sequence $\mathbf{P}$:

$$P^{\lambda}_{Pse-PSSM} = [\overline{E}_1, \overline{E}_2, \dots, \overline{E}_{20}, G^{\lambda}_1, G^{\lambda}_2, \dots, G^{\lambda}_{20}] \tag{14}$$

where $\overline{E}_j$ and G^{λ}_j are defined as follows:

$$\overline{E}_j = \frac{1}{L}\sum_{i=1}^{L} E_{i\to j} \qquad (j = 1, 2, \dots 20) \tag{15}$$

$$G^{\lambda}_j = \frac{1}{L-\lambda}\sum_{i=1}^{L-\lambda} [E_{i\to j} - E_{(i+\lambda)\to j}]^2 \qquad (j = 1, 2, \dots 20) \tag{16}$$

λ is chosen such that ($\lambda < L$). Since the shortest protein in our dataset is 50 amino acids long, we considered all $\lambda \in (0, \dots, 49)$, and the performance of each encoding was evaluated separately.

2.3.2 Split amino acid composition (SAAC)

The concept of SAAC was first reported by Hayat et al. [Hay12]. The motivation behind this concept is that sometimes the most important information is concealed in fragments, and when calculating the AAC for the whole sequence, such information may be masked by noisy, irrelevant information. The SAAC is the sequence encoding used by iMem-2LSAAC, a state-of-the-art predictor of membrane proteins [AHJ18].

In SAAC, a protein sequence is divided into segments, and the AAC is computed for each segment separately. Here, we followed the same partitioning described for iMem-2LSAAC [AHJ18]: the sequence is divided into three sections, namely, the first 25 amino acids of the N-terminus, the last 25 amino acids of the C-terminus, and the region between these sections. Each protein is then represented by a vector of size 60, as follows:

$$SAAC(P) = [c_1^N, c_2^N, \dots c_{20}^N, c_1, c_2, \dots c_{20}, c_1^C, c_2^C, \dots c_{20}^C] \tag{17}$$

where c_i^N, c_i, and c_i^C are the normalized occurrence frequencies of the i^{th} amino acid in the N-terminus, between the two termini, and C-terminus segments, respectively.

2.4 Machine learning algorithms

2.4.1 K-nearest neighbor (KNN)

The KNN classification algorithm is simple and effective. It is a type of instance-based learning, where all computations are deferred until prediction time. The KNN algorithm assigns a class to an unclassified object X based on the class represented by the majority of its KNNs in the training set vectors. If K = 1, the class of object X will be the class of its nearest neighbor. The choice of K is key to the quality of the KNN prediction engine; we found that the performance started to deteriorate when $K > 10$. We also found that fusing the results of 10 individual classifiers, where $K \in (1, \dots, 10)$ through majority voting, achieved the highest accuracy and was adopted for the KNN models. We applied the KNN algorithm as implemented by the *class* library in R (version 7.3-15).

2.4.2 Optimized evidence-theoretic k-nearest neighbor (OET-KNN)

The OET-KNN algorithm is a modification of the traditional KNN algorithm and has been shown to be highly powerful in statistical prediction [Den95]. It has been used by one of the most powerful membrane predictors, MemType-2L. The OET-KNN algorithm is based on the Dempster-Shafer theory of belief functions [Den95], wherein each neighbor in a pattern to be classified is regarded as evidence supporting certain hypotheses concerning the class membership of that object. As with the KNN algorithm, any constructed OET-KNN model is an ensemble of multiple OET-KNN classifiers, each with different values of $K \in (1, \ldots, 10)$. The final class was determined through majority voting. We used the OET-KNN algorithm as implemented in R by the *evclass* library (version 1.1.1).

2.4.3 Support vector machine (SVM)

SVMs are a powerful machine learning tool used in many biological prediction tools, such as in [MCZ14] and [AHJ18]. SVMs aim at solving classification problems by finding appropriate decision boundaries between different classes. In relation to nonlinearly separable data, the kernel trick can be used to transform nonlinear data into a higher-dimensional space where optimal boundaries can be found in an efficient, less computationally expensive process compared to the explicit computations of the coordinates. We used an SVM with a radial basis function (RBF) kernel as implemented by the R *e1071* library (version 1.6-8). The best combination of the C and γ parameters was determined by utilizing a grid search approach.

2.4.4 Gradient-boosting machine (GBM)

GBMs are a machine learning technique that produces a strong model by assembling weak prediction models, usually decision trees. They use gradient boosting by iteratively training new models based on the weak points of the previous models. While not commonly applied in biological predictions, GBMs have been demonstrated to be one of the most powerful techniques on the popular machine learning competition website Kaggle (kaggle.com). Here, we used the *xgboost* library (version 0.81.0.1), which is a fast and efficient implementation of the gradient-boosting framework in R.

2.5 Ensemble classifier

2.5.1 All voting

Let C_i^{ML} be a classifier built using the machine learning algorithm ML $\in$ {KNN, OET-KNN, SVM, GBM}, in which the protein samples are represented by Pse-PSSM, with $\lambda = i$ and $i \in (0, \ldots, 49)$; each classifier is constructed as described in Section 2.4. In addition,

let $C_{i,k}^{ML}$ be a classifier built using the machine learning algorithm ML $\in$ {KNN, OET-KNN} in which the protein samples are represented by Pse-PSSM, with $\lambda = i$ and $i \in (0, \ldots, 49)$; and the parameter K that refers to number of neighbors equals k and $k \in (1, \ldots, 10)$.

In *all voting*, we evaluated the following six different ensembles:

SVM-based ensemble: obtains the results from 50 SVM-based classifiers ($C_0^{SVM}, C_1^{SVM} \ldots C_{49}^{SVM}$) and combines them through a voting mechanism, where the class that receives the most votes is chosen by the ensemble classifier.

GBM-based ensemble: obtains the results from 50 GBM-based classifiers ($C_0^{GBM}, C_1^{GBM} \ldots C_{49}^{GBM}$) and combines them through the same voting mechanism as above.

KNN V50-based ensemble: obtains the results from 50 KNN-based ensemble classifiers ($C_0^{KNN}, C_1^{KNN} \ldots C_{49}^{KNN}$) and combines them through the same voting mechanism.

KNN V500-based ensemble: obtains the results from 500 KNN-based classifiers (50 for different values of λ multiplied by 10 for different values of K; $C_{0,1}^{KNN}, C_{0,2}^{KNN} \ldots C_{49,10}^{KNN}$) and combines them through the same voting mechanism.

OET-KNN V50-based ensemble: obtains the results from 50 OET-KNN-based ensemble classifiers ($C_0^{OET-KNN}, C_1^{OET-KNN} \ldots C_{49}^{OET-KNN}$) and combines them through the same voting mechanism.

OET-KNN V500-based ensemble: obtains the results from 500 OET-KNN-based classifiers (50 for different values of λ multiplied by 10 for different values of K; $C_{0,1}^{OET-KNN}, C_{0,2}^{OET-KNN} \ldots\ C_{49,10}^{OET-KNN}$) and combines them through the same voting mechanism; this is the MemType-2L approach [CS07].

2.5.2 Selective voting

For each ensemble in *all voting*, rather than fusing the predictions from all of the individual predictors, here, the optimal subset of predictions (i.e., the output of the constituent classifiers) is selected so that they have minimal redundancy and maximal relevance with the target class. To accomplish this task, we first ranked the features using the minimum redundancy maximum relevance (mRMR) algorithm [PLD05], as implemented by the `R` *mRMRe* library (version 2.1.0), and then utilized incremental feature selection [HSW$^+$10] to choose the optimal subset.

To quantify both the relevance and redundancy, *mRMRe* uses a linear approximation based on correlation such that mutual information (MI) between two variables c_i, c_j is estimated as:

$$MI(c_i, c_j) = -\frac{1}{2} ln(1 - \rho(c_i, c_j)^2) \tag{18}$$

ρ is the Cramer's V coefficient between c_i and c_j.

Let y be the target class and $X = (c_1, c_2, \ldots, c_n)$ be the set of n input features, i.e., the set of constituent classifiers output in *all voting*. The mRMR method ranks the features in X by maximizing the MI with y (maximum relevance) and minimizing the average MI with all the previously selected variables (minimum redundancy). A list of selected features, denoted by S, is initialized with c_i, the feature with highest MI with the target variable such that:

$$c_i = \underset{c_i \in X}{\arg\max}\, MI(c_i, y) \tag{19}$$

Next, another feature, c_j, is added to S by choosing the feature that has the highest relevance with the output variable and the lowest redundancy with the previously selected features, utilizing the mutual information difference (MID) scheme:

$$c_j = \max_{c_j \in \Omega S} \left[MI(c_j, y) - \frac{1}{|S|} \sum_{c_i \in S} MI(c_j, c_i) \right] \tag{20}$$

ΩS denotes the set of features that are not yet added to S. This is continued until all of the features in X are added to S:

$$S = (c'_1, c'_2, \ldots, c'_n) \tag{21}$$

c'_i denotes the feature with the i^{th} rank. Next, we utilized incremental feature selection [HSW+10] to choose the optimal subset. Incremental feature selection constructs n sets by adding one component at a time in an ascending order, with the i^{th} given as:

$$s_i = \{c'_1, c'_2 \ldots c'_i\} \qquad (1 \leq i \leq n) \tag{22}$$

The set with the highest accuracy is then selected for *selective voting*.

2.6 Performance measurement

The performances of the different prediction models were evaluated by leave-one-out cross-validation (LOOCV), in which each sample in the training dataset is predicted based on the rules derived from all of the other samples except the one being predicted; this procedure is repeated so that each sample is used once for validation.

The LOOCV approach was applied to evaluate the state-of-the-art methods of the all-type membrane predictors iMem-2LSAAC [AHJ18] and MemType-2L [CS07], the LOOCV approach was chosen here because goes through all the samples in the training set, and its performance does not vary with different runs.

Furthermore, we evaluated the performance of the model that achieved the highest performance during LOOCV using an independent testing set and compared it to those achieved by the models built with the state-of-the-art methods. Four main evaluation metrics were considered: the sensitivity, specificity, accuracy, and MCC. The sensitivity indicates the proportion of positive samples that are correctly identified.

$$Sensitivity = \frac{TP}{TP + FN} \tag{23}$$

The specificity measures the proportion of negative samples that are correctly identified.

$$Specificity = \frac{TN}{TN + FP} \tag{24}$$

The accuracy is the number of correct predictions divided by the total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{25}$$

The MCC measures the quality of a binary classifier 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.

$$MCC = \frac{(TP \times TN - FP \times FN)}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \tag{26}$$

3 Experimental design

The first experiment encodes protein sequences using different methods and uses the generated vectors as input to train different models based on the KNN, OET-KNN, SVM and GBM algorithms; the performances of different models are evaluated on the training set using LOOCV. The second experiment evaluates the two ensemble approaches, *all voting* and *selective voting*, and compares their performances. The third experiment evaluates the performances of the HMMTOP [TS01], TMHMM [KLvHS01], TOPCONS2 [TPS$^+$15] and PRED-TMBB2 [TEB16] topology prediction tools for detecting all membrane types. Finally, the last experiment integrates the prediction achieved by the best-performing topology prediction tool with the best-performing ensemble in the second experiment; we refer to this integrative approach as *TooT-M*.

In all the abovementioned experiments, only the training set is used to choose the best model/tool. The best-performing method in all of the experiments is chosen as our membrane predictor, and ultimately, its performance is tested on the independent testing set and compared to that achieved by the state-of-the-art methods.

4 Results and discussion

4.1 Evaluation of different protein encodings

The LOOCV performances of the baseline encodings AAC, PAAC, and PseAAC, in addition to SAAC, which is utilized by iMem-2LSAAC [AHJ18], and the Pse-PSSM utilized by MemType-2L [CS07] on different machine learning algorithms are illustrated in Table 2. Only the Pse-PSSMs where $\lambda \in (0, 1, 2)$ are presented here; the rest have comparable performances and are found in Appendix A.

The encoding extraction techniques can be divided into two primary groups: techniques that extract features solely from a protein sequence, such as AAC, PAAC, PseAAC, and SAAC, and the Pse-PSSM technique that incorporates evolutionary information. Among those techniques that extract features from the protein sequence alone, PseAAC in combination with GBM achieved the highest performance, with an overall validation accuracy of 80.60%, followed by PAAC and SVM, for which the overall accuracy reached 80.28%. The SAAC encoding method used by iMem-2LSAAC [AHJ18] was not superior to the other feature extractors, and it reached its highest overall accuracy (80.00%) with the GBM model.

The encoding technique that integrates evolutionary information in the form of Pse-PSSMs for all $\lambda \in (0, \ldots, 49)$ consistently achieved higher accuracy by an average of 11% relative to the methods that rely solely on the protein sequences of individual samples. The highest accuracy reached 89.70% and was achieved by OET-KNN, where the protein samples were encoded using Pse-PSSM $\lambda = 0$. However, when the protein samples were encoded using Pse-PSSM $\lambda \in (1, \ldots, 49)$, the SVM-based models outperformed the models based on the OET-KNN, KNN, and GBM algorithms.

Table 2: LOOCV performance of the individual models

Encoding	**ML Algorithm**	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
AAC	OET-KNN	71.34	81.08	76.28	0.5271
	KNN	75.72	74.87	75.29	0.5058
	SVM	70.96	83.47	77.30	0.5492
	GBM	71.86	83.75	77.89	0.5606
PseAAC	OET-KNN	73.05	81.38	77.27	0.5465
	KNN	74.24	79.38	76.84	0.5370
	SVM	70.59	83.98	77.37	0.5511
	GBM	74.99	86.07	80.60	0.6149
PAAC	OET-KNN	68.94	72.09	70.53	0.4105
	KNN	72.96	66.26	69.57	0.3930
	SVM	76.15	84.22	80.24	0.6060
	GBM	71.33	85.01	77.84	0.5661
SAAC	OET-KNN	66.63	72.88	69.80	0.3960
	KNN	69.75	68.81	69.28	0.3856
	SVM	**72.51**	**85.85**	**79.27**	**0.5895**
	GBM	73.90	85.95	80.00	0.6034
Pse-PSSM, $\lambda = 0$	OET-KNN	86.57	92.75	89.70	0.7953
	KNN	85.22	90.44	87.86	0.7580
	SVM	83.23	90.05	86.68	0.7350
	GBM	83.41	90.45	86.98	0.7409
Pse-PSSM, $\lambda = 1$	OET-KNN	85.92	91.79	88.89	0.7788
	KNN	85.89	89.06	87.50	0.7501
	SVM	86.75	92.22	89.52	0.7912
	GBM	85.00	92.19	88.64	0.7744
Pse-PSSM, $\lambda = 2$	OET-KNN	85.51	91.90	88.75	0.7762
	KNN	85.65	88.28	86.98	0.7397
	SVM	86.83	92.06	89.48	0.7904
	GBM	84.86	91.72	88.34	0.7682
Pse-PSSM, $\lambda = 3$	OET-KNN	84.69	91.44	88.11	0.7636
	KNN	84.97	87.92	86.47	0.7295
	SVM	86.97	91.61	89.32	0.7871
	GBM	85.01	91.74	88.42	0.7697
Pse-PSSM, $\lambda = 4$	OET-KNN	85.46	91.37	88.45	0.7701
	KNN	85.44	88.41	86.95	0.739
	SVM	86.87	91.85	89.39	0.7886
	GBM	85.71	92.41	89.11	0.7835
Pse-PSSM, $\lambda = 5$	OET-KNN	85.17	91.32	88.29	0.7668
	KNN	85.6	88.07	86.85	0.7371
	SVM	86.82	92.26	89.58	0.7925
	GBM	85.1	92.08	88.63	0.7742

This table shows microaverage LOOCV performance of the different protein encodings on different machine learning algorithms. The SAAC with SVM, highlighted in bold, reflects the LOOCV performance of the iMem-2LSAAC method [AHJ18] on *DS-M*. Only the Pse-PSSMs where $\lambda \in (0, \ldots, 5)$ are shown here; the complete performance of all the Pse-PSSMs ($\lambda \in (0, \ldots, 49)$) can be found in Appendix A.

4.2 Evaluation of the ensemble techniques

The performance of the first ensemble approach, *all voting*, on the training dataset is presented in Table 3. Since the data are balanced, we focused on the accuracy when comparing the performance of the different models. Among the six ensembles in *all voting*, the SVM-based ensemble achieved the highest accuracy of 90.15%. The OET-KNN V500 ensemble, which reflects the performance of MemType-2L [CS07] on *DS-M*, achieved the second highest accuracy of 89.86%.

Table 3: Performances of the *all voting* ensemble classifiers on the training dataset

Algorithm	Sensitivity	Specificity	Accuracy	MCC
OET-KNN V500	**85.10**	**94.51**	**89.86**	**0.8004**
OET-KNN V50	85.61	93.57	89.64	0.7950
KNN V500	85.50	91.77	88.68	0.7747
KNN V50	86.19	90.40	88.32	0.7669
SVM	86.48	93.72	90.15	0.8047
GBM	84.52	93.32	88.98	0.7820

all voting with OET-KNN V500, highlighted in bold, reflects the LOOCV performance of the MemType-2L method on *DS-M*.

To choose the optimal feature set for *selective voting*, we tested the mRMR top-ranked c ($1 \leq c \leq 50$) features incrementally by adding one feature at a time to the OET-KNN V50, KNN V50, SVM, and GBM models, and the top-ranked c ($1 \leq c \leq 500$) features on the OET-KNN V500 and KNN V500 models. The optimal feature set is the one with the highest accuracy. As observed from Figure 14, the accuracy peaked when the number of top-ranked components were 3, 5, 15, and 11 for the OET-KNN V50-, KNN V50-, SVM-, and GBM-based ensembles, respectively. In addition, the optimal number of features for the OET-KNN V500 and KNN V500 ensembles were 20 and 21, respectively, as shown in Figure 15; the performance started to deteriorate as more votes were counted. The detailed performances of the optimal feature set are presented in Table 4.

The results show that the ensemble models outperform their constituent classifiers, and the *selective voting* ensemble approach outperforms the *all voting* approach. Generally, the ensemble works best when the individual classifiers comprising the ensemble are both accurate and have low correlation [OS96] [KV95]. The superiority of *selective voting* over *all* *voting* is due to the mRMR method's ability to choose the models that have low correlations among each other and high correlation with the target class (i.e., most accurate) and to the incremental feature selection's ability to select the optimal set that reduces the noise and increases the ensemble classifier's distinctive power. An interesting observation to note here is that while the individual SVM and GBM classifiers generally provided higher performances than those of the OET-KNN and KNN classifiers, the latter leveraged more from the *selective voting* ensemble. This suggests that the predictions from the OET-KNN and KNN classifiers are less consistent (i.e., they make errors in different parts of the input space) and are therefore better candidates for the ensemble than the SVM and GBM classifiers.

The best performance in all methods was achieved by *selective voting* with the OET-KNN V500 ensemble, where the overall accuracy reached 91.31%, which is 1.67% higher than what the MemType-2L method (OET-KNN V500 with *all voting*) achieved. Because it achieved the best performance, the *selective voting* approach with the OET-KNN V500 method is utilized in the integrative approach *TooT-M*.

Table 4: Performances of the *selective voting* ensemble classifiers on the training dataset

Algorithm	Sensitivity	Specificity	Accuracy	MCC
OET-KNN V500	**88.99**	**94.00**	**91.53**	**0.8314**
OET-KNN V50	86.58	94.43	90.56	0.8133
KNN V500	89.01	93.63	91.35	0.8280
KNN V50	86.55	91.92	89.27	0.7863
SVM	87.12	93.72	90.46	0.8107
GBM	85.30	93.45	89.44	0.7909

Selective voting with OET-KNN V500, highlighted in bold, refers to the method that achieved the highest MCC and is the method utilized in *TooT-M*

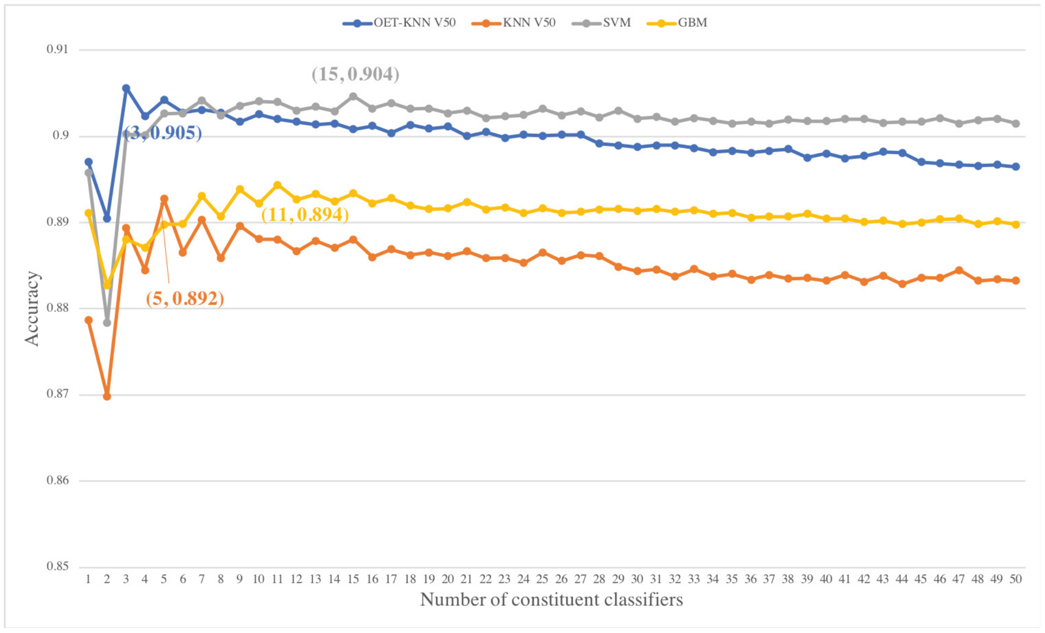


Figure 14: Choice of the optimal constituent classifiers among 50 classifiers

In the pair (x, y), x refers to the number of top-ranked components in the optimal feature set, and y refers to the achieved accuracy using those x components. The accuracy peaked when the number of top-ranked components were 3, 5, 15, and 11 for the OET-KNN V50-, KNN V50-, SVM-, and GBM-based ensembles, respectively.

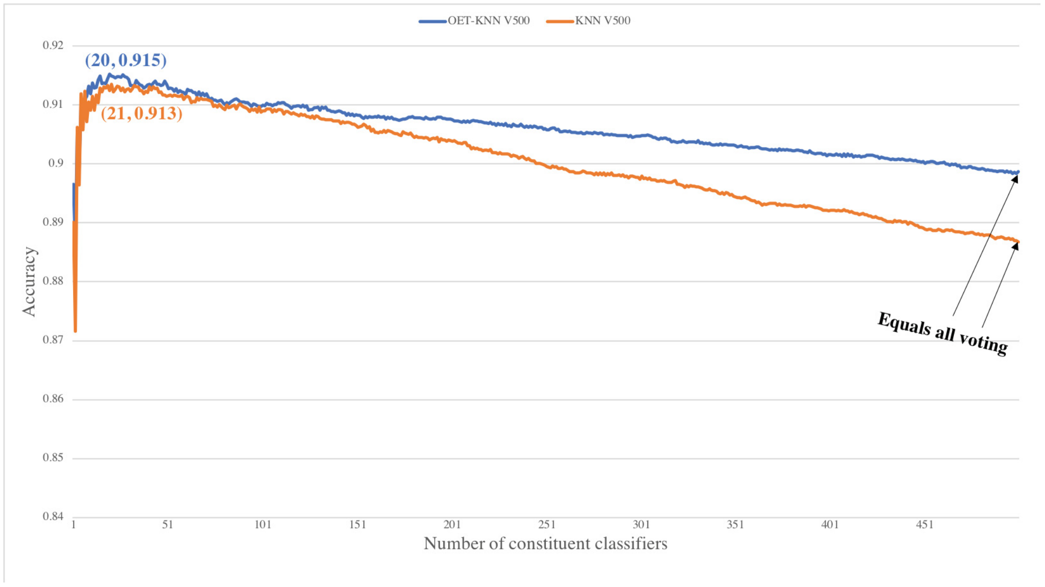


Figure 15: Choice of the optimal constituent classifiers among 500 classifiers

In the pair (x, y), x refers to the number of top-ranked components in the optimal feature set, and y refers to the achieved accuracy using those x components. The optimal numbers of features for the OET-KNN V500 and KNN V500 ensembles were 20 and 21, respectively. The performance started to deteriorate as more votes were accounted for. Overall, the results suggest that the *selective voting* approach outperforms the *all voting* approach.

4.3 Evaluation of transmembrane topology prediction tools

The performances of HMMTOP [TS01], TMHMM [KLvHS01], TOPCONS2 [TPS+15] and PRED-TMBB2 [TEB16] on the *DS-M* training set are shown in Table 5. Based on our analysis in Section 2.1, we expected the topology prediction tools to fail to predict at least 20% of the membrane proteins because they do not contain TMSs; the results reported here confirm this hypothesis. The transmembrane topology reached a maximum sensitivity of 72%. This finding further highlights the importance of building a model to predict all membrane types and that transmembrane topology tools disregard surface-bound proteins and thus fail to recognize more than 20% of membrane proteins. Nevertheless, a very attractive aspect here is the exceptionally high specificity (true negative rate) in TOPCONS2, which is due its ability to distinguish signal peptides from transmembrane regions [TGB+18]. This property means that the confidence in the positive prediction is high; thus, this aspect is exploited in *TooT-M*.

Table 5: Transmembrane topology prediction performance on the training dataset

Tool	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
HMMTOP	72.71	84.60	78.73	0.5777
TOPCONS2	**69.86**	**99.77**	**85.01**	**0.7318**
TMHMM	68.61	97.14	83.06	0.6878
PRED-TMBB2	41.73	55.48	48.70	-0.0281

TOPCONS2, highlighted in bold, is the tool that achieved the highest MCC and is the method utilized in *TooT-M*.

4.4 Performance of ***TooT-M***

The integrative approach *TooT-M* combines the best models from both the transmembrane topology tools (TOPCONS2) and the all-type membrane predictors (*selective voting* OET-KNN V500) through weighted voting. In weighted voting, a positive vote from TOPCONS2 is trusted and multiplied by the number of constituent classifiers in the *selective voting* OET-KNN V500 ensemble minus one; that is, the OET-KNN V500 *selective voting* prediction is transformed to positive *if and only if* there is at least one constituent classifier that agrees with the positive prediction of TOPCONS2. Among all the tested weights, this approach helped enhance the sensitivity without negatively impacting the specificity.

Table 6 shows the LOOCV performance of *TooT-M*. Compared to the *selective voting* OET-KNN V500 ensemble, the sensitivity (true-positive rate) was enhanced by 2.76% and the specificity was enhanced by 1.35%. Overall, the accuracy increased by 2%, and the MCC was boosted by 4%.

Table 6: *TooT-M* LOOCV performance

Method	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
Selective voting OET-KNN V500	89.01	93.63	91.35	0.8280
TOPCONS2	69.86	99.77	85.01	0.7318
TooT-M	**91.47**	**94.90**	**93.21**	**0.8645**

This table shows the LOOCV performance of *TooT-M*, which integrates the predictions from the constituent classifiers of the *selective voting* OET-KNN V500 ensemble and TOPCONS2 through weighted voting.

4.5 Comparison with the state-of-the-art methods

Here we compare the performance of *TooT-M* to the state-of-the-art methods in three settings:

1. All the methods are trained on the *DS-M* training set, and their performances are evaluated on the *DS-M* testing set.

2. The *TooT-M* method is trained on the dataset obtained by the iMem-2LSAAC authors (DS1), and its performance is compared with the reported performance of iMem-2LSAAC [AHJ18] on the same dataset.

3. The *TooT-M* method is trained on the dataset provided by Chou and Shen [CS07] (DS2), and its performance is compared to the reported performance of MemType-2L [CS07] on the same dataset.

As illustrated in Figure 16 and indicated in Table 7, the integrative approach outperformed all of the other methods in terms of sensitivity, specificity, accuracy, and MCC. Similarly, as shown in Table 8, it outperformed Mem-2LSAAC [AHJ18] in terms of specificity, accuracy,

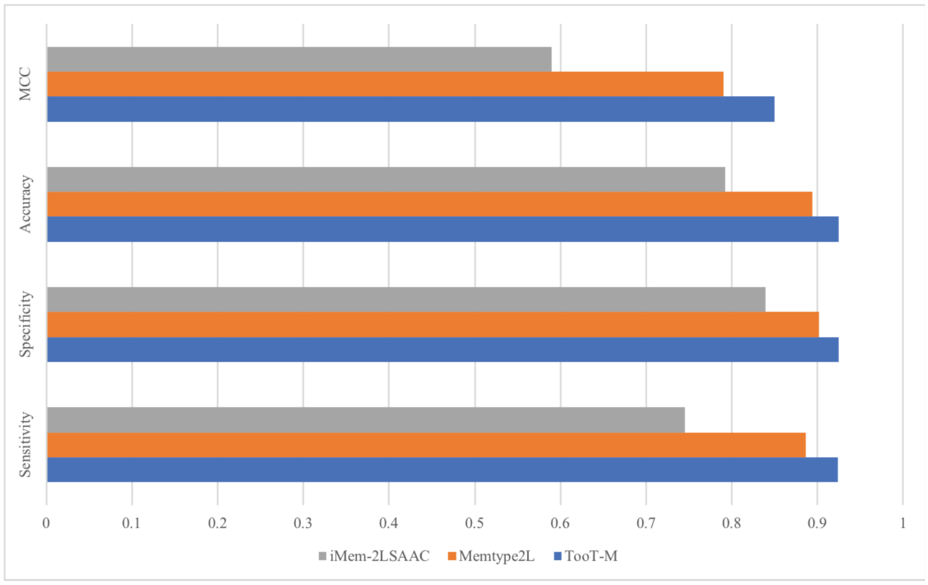


Figure 16: Comparison with other state-of-the-art methods on the *DS-M* dataset

and MCC, while still keeping the sensitivity credible. It also outperformed MemType-2L [CS07] in terms of sensitivity, accuracy, and MCC, while achieving a similar specificity, as shown in Table 9.

Table 7: Comparison with other state-of-the-art methods on the *DS-M* dataset

Method	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
TooT-M	**92.41**	**92.5**	**92.46**	**0.85**
MemType-2L [CS07]	88.67	90.19	89.44	0.79
iMem-2LSAAC [AHJ18]	74.52	83.9	79.27	0.59

This table compares the performance of the integrative approach with other state-of-art methods on the *DS-M* dataset. The highest performance in each metric is highlighted in bold. *TooT-M* outperformed the state-of-the-art methods across all metrics.

Table 8: Comparison with the iMem-2LSAAC predictor on the DS1 dataset

Method	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
TooT-M	98.09	**96.80**	**97.43**	**0.94**
iMem-2LSAAC	**98.23**	91.17	94.61	0.89

This table compares the performance of *TooT-M* with the state-of-art iMem-2LSAAC predictor [AHJ18] on the same dataset, DS1. The best performance for each metric is highlighted in bold. *TooT-M* achieved a higher specificity, accuracy and MCC than iMem-2LSAAC.

Table 9: Comparison with the MemType-2L predictor on the DS2 dataset

Method	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
TooT-M	**92.71**	**94.4**	**93.57**	**0.87**
MemType-2L	91.00	**94.4**	92.7	0.85

This table compares the performance of *TooT-M* with the state-of-art MemType-2L predictor [CS07] on the same dataset, DS2. The best performance for each metric is highlighted in bold. *TooT-M* achieved a higher sensitivity, accuracy and MCC than MemType-2L and the same specificity.

5 Conclusion

We curated a new membrane protein benchmark dataset that contains all types of membrane proteins, including surface-bound proteins. We demonstrated the limitation of using only transmembrane topology prediction tools to predict all types of membrane proteins, as they detect only IMPs and miss surface-bound proteins, which account for approximately 20% of membrane protein data. Furthermore, we evaluated the performances of different protein-encoding techniques, including those employed by the state-of-the-art membrane predictors with different machine learning algorithms. The experimental results obtained by cross-validation and independent testing suggest that applying an integrative approach that combines the results of transmembrane topology prediction and Pse-PSSM OET-KNN predictors yields the best performance. *TooT-M* achieved a 92.46% accuracy in independent testing, compared to the 89.44% and 79.27% accuracies achieved by the state-of-the-art methods MemType-2L [CS07] and iMem-2LSAAC [AHJ18], respectively.

Chapter 5

Transporter protein detection

The main focus of this chapter is the second research objective:

O2: To improve the computational tools for predicting de novo transporters, relying only on the primary protein sequence

Some of the contents of this chapter have been published in *BMC Bioinformatics*: M. Alballa, G. Butler, TooT-T: Discrimination of transport proteins from non-transport proteins, *BMC Bioinformatics*, 21.3 (2020): 1-10.

This chapter is organized as follows: Section 1 provides an introduction to the work and highlights the main contributions we make in the chapter. Section 2 gives an overview of the proposed tool (*TooT-T*). Section 3 lists the material and methods utilized in *TooT-T* and introduces a new method of encoding a protein sequence, *psi-composition*, that combines the traditional compositions with evolutionary information obtained from a PSI-BLAST search. Section 4 presents and discusses the results, and Section 4.4 compares the results achieved by *TooT-T* with those obtained by the state-of-the-art tools. Finally, Section 5 concludes the chapter.

1 Introduction

Transporters control the movement of molecules across the membrane so that essential molecules such as sugars and amino acids enter the cell while waste compounds leave the cell. It is estimated that membrane transport proteins encode 2% to 16% of ORFs in prokaryotic and eukaryotic genomes, highlighting the importance of transporters in all living species [RP05].

While many membrane proteins sequences are known due to the large number of recent genome projects, their structures and functions remain poorly characterized and understood. This deficiency is related to the immense effort necessary to characterize these proteins because of their structural flexibility and instability, which create challenges at many levels, including crystallization, expression, and structure solution. This imbalance between the number of available sequences and experimentally characterized sequences has created many obstacles in the advancement of biology and drug discovery. Therefore, there is a need for advanced computational techniques that can utilize the sequence information alone to distinguish membrane transporter proteins. These novel techniques can then be used to direct new experiments and offer clues about protein function.

The findings from previous studies on transporter predictions can be summarized as follows: an SVM achieved superior performance compared to other machine learning algorithms [LVY+15] [LLX+16] [HPO+19]. Moreover, PSSM profiles are a highly accurate encoding method for demonstrating evolutionary information within protein sequence functional classifications [LVY+15] [MCZ14] [HGS+15].

This work focuses on distinguishing membrane transporter proteins from other non-transporter proteins. The main contributions of this work can be summarized as follows:

- We explore the practicality of using traditional homology search techniques to detect transporter proteins.
- We compare the performances of various discriminators/encodings on SVM models and introduce a new encoding, called *psi-composition*, which shows superior performance compared to all of the other examined encodings.
- We propose a new tool, *TooT-T*, that employs an ensemble classifier that is trained to optimally combine the predictions obtained from homology annotation transfer and psi-composition-based models to determine the final prediction. The ensemble exploits the low correlation between the predictions obtained by various methods to build a more robust classifier. The proposed model outperforms all of the state-of-the-art

methods that rely on the protein sequence alone, with overall accuracies of 97.02% and 97.28% and MCCs of 0.92 and 0.93 in cross-validation and independent testing, respectively.

2 ***TooT-T* overview**

TooT-T utilizes an ensemble classifier that combines the results generated by two distinct methods, namely, homology annotation transfer and machine learning, to detect transporter proteins. First, given a query protein Q, a traditional homology search of the TCDB is performed using BLAST. A query is predicted as a transporter if a hit is found using three predetermined sets of thresholds. The three predictions are delivered to the ensemble. Then, three variations of psi-composition encoding (psiAAC, psiPAAC, and psiPseAAC) are computed and input into their respective trained SVM models. Finally, the trained ensemble meta-model predicts the final class as a transporter T or non-transporter NT. Figure 17 delineates an overview of the *TooT-T* prediction steps. The motivation behind this selection and descriptions of each step are presented in the following sections.

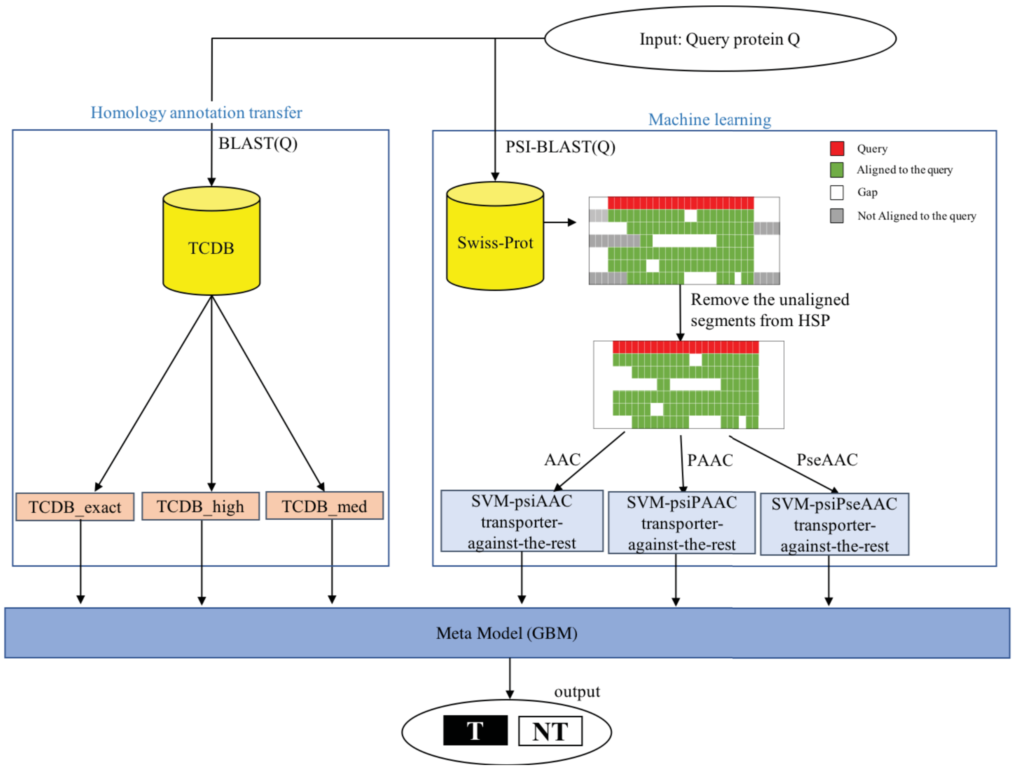


Figure 17: *TooT-T* overview

When a new query protein is input into *TooT-T*, the class of the query is predicted by the six base classifiers: three from SVM models based on psiAAC, psiPAAC, and psiPseAAC encoding, and three from annotation transfer by homology utilizing different thresholds (TCDB_exact, TCDB_high, and TCDB_med). The six predictions are then input into the meta-classifier, which outputs the final prediction.

3 Materials and methods

3.1 Dataset

A new benchmark membrane transporter dataset from the `Swiss-Prot` database (June 2018 release) was collected. The initial dataset was constructed as follows:

Protein sequences that belong to the *transporter* class were retrieved using the following search query:

```
locations:(location:membrane)
goa:("transporter activity [5215]") AND reviewed:yes
```

This query searches for proteins that have the `GO:0005215 transporter activity` `GO` MF annotation. This `GO` MF was chosen here because it is directly related to the actual function of the protein rather than the general process in which it is involved.

Protein sequences that do not belong to the transporter class but are located in the membrane were retrieved as *non-transporters* using the following search query:

```
locations:(location:membrane)
NOT goa:("transporter activity [5215]")
AND reviewed:yes
```

The initial set was then filtered to attain the best-quality dataset by adhering to the following criteria:

- **Step 1:** Protein sequences that have evidence "inferred from homology" for the existence of a protein were removed.
- **Step 2:** Protein sequences that are annotated with multiple functions (e.g., transporters and enzymes) were removed.
- **Step 3:** Protein sequences that have no GO MF annotation or annotation based only on computational evidence (IEA) were eliminated.
- **Step 4:** Protein sequences with more than 60% pairwise sequence identity were removed via the CD-HIT [LG06] program to avoid any homology bias.

Details about the number of samples attained by each step in the curation process are presented in Figure 18. The final dataset was established after Step 4 and was randomly partitioned (stratified by class) into a training set (90% of the data) and a testing set (10% of the data), as presented in Table 10. Since the testing set was kept aside until the final testing phase and was not used during the training or model selection, we refer to it as independent testing set.

Class	**Training**	**Testing**	**Total**
Transporter	2,002	222	2,224
Non-transporter	5,943	661	6,604
Total	7,945	883	8,828

Table 10: Transporter membrane dataset *DS-T*

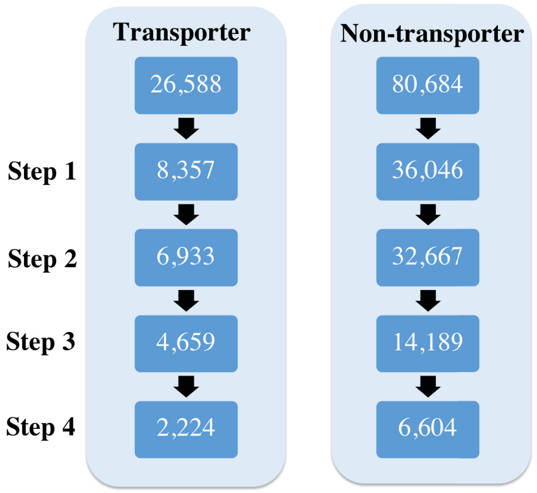


Figure 18: Membrane protein curation process

This figure shows details on the number of samples during each step of the curation process. Step 1: Protein sequences that had evidence for the existence of a protein "inferred from homology" were removed. Step 2: Protein sequences annotated with multiple functions (e.g., as transporters and enzymes) were set aside for further examination. Step 3: Protein sequences with no `GO` MF annotation or annotation based only on computational evidence (IEA) were eliminated. Step 4: Protein sequences with more than 60% pairwise sequence identity were removed.

3.2 Position-specific iterative encodings

A PSI-BLAST search [AMS+97] (3 iterations, e-value cutoff 0.001) was performed on a sample protein sequence using a modified version of the `Swiss-Prot` database (June 2018 release) to find homologous sequences. The modified `Swiss-Prot` database did not include the exact hits of the test sequences. Regions in the database hit sequences that were not aligned with the query protein were discarded. The query protein (Q) and the aligned regions of its hits $(h_1, h_2, ..., h_n)$ were then used to compute the position-specific iterated AAC (psiAAC), PAAC (psiPAAC), and PseAAC (psiPseAAC) as described in the following sections.

Position-specific iterated amino acid composition (psiAAC)

The AAC of the query protein (Q) and each of its filtered hits $(h_1, h_2, \ldots, h_n)$ were calculated separately as the fractions of all 20 natural amino acids:

$$c_i = \frac{F_i}{L} \qquad i = (1, 2, 3, ...20) \tag{27}$$

where F_i is the frequency of the i^{th} amino acid and L is the length of the sequence. The AAC is represented as a vector of size 20:

$$AAC(P_x) = [c_1, c_2, c_3, ..., c_{20}] \qquad x \in (Q, h_1, h_2 \ldots, h_n) \tag{28}$$

where c_i is the composition of the i^{th} amino acid. Then, the mean of individual AAC compositions represents the psiAAC for Q and was computed as:

$$AAC_{psi}(Q) = \frac{1}{n+1} \sum AAC(P_x) \qquad x \in (Q, h_1, h_2 \ldots, h_n) \tag{29}$$

Position-specific iterated pair amino acid composition (psiPAAC)

Similarly, the individual PAAC descriptors for the query protein (Q) and each of its filtered hits $(h_1, h_2, \ldots, h_n)$ were calculated as follows:

$$d_{i,j} = \frac{F_{i,j}}{L-1} \qquad i, j = (1, 2, 3, ...20) \tag{30}$$

where $F_{i,j}$ is the frequency of the i^{th} and j^{th} amino acids as a pair (dipeptide) and L is the length of the sequence. Similar to the AAC, the PAAC is represented as a vector of size 400 as follows:

$$PAAC(P_x) = [d_{1,1}, d_{1,2}, d_{1,3}, ..., d_{20,20}] \qquad x \in (Q, h_1, h_2 \ldots, h_n) \tag{31}$$

where $d_{i,j}$ is the dipeptide composition of the i^{th} and j^{th} amino acids. The mean of individual PAAC compositions represents the psiPAAC for Q and was computed as follows:

$$PAAC_{psi}(Q) = \frac{1}{n+1} \sum PAAC(P_x) \qquad x \in (Q, h_1, h_2 \ldots, h_n) \tag{32}$$

Position-specific iterated pseudo amino acid composition (psiPseAAC)

The PseAAC is a combination of the 20 components of the conventional AAC and a set of sequence-order correlation factors that incorporate certain biochemical properties, originally proposed by Chou [Cho01]. Given a protein sequence of length L:

$$R_1R_2R_3R_4...R_L \tag{33}$$

a set of descriptors called sequence-order-correlated factors are defined as:

$$\begin{cases} \theta_1 = \dfrac{1}{L-1}\displaystyle\sum_{i=1}^{L-1}\Theta(R_i, R_{i+1}) \\ \theta_2 = \dfrac{1}{L-2}\displaystyle\sum_{i=1}^{L-2}\Theta(R_i, R_{i+2}) \\ \theta_3 = \dfrac{1}{L-3}\displaystyle\sum_{i=1}^{L-3}\Theta(R_i, R_{i+3}) \\ \quad . \\ \quad . \\ \quad . \\ \theta_\lambda = \dfrac{1}{L-\lambda}\displaystyle\sum_{i=1}^{L-\lambda}\Theta(R_i, R_{i+\lambda}) \end{cases} \tag{34}$$

The parameter λ is chosen such that ($\lambda < L$). A correlation function is given by:

$$\begin{aligned}\Theta(R_i, R_j) = \frac{1}{3}\big\{[H_1(R_j) - H_1(R_i)]^2 + [H_2(R_j) - H_2(R_i)]^2 \\ +[M(R_j) - M(R_i)]^2\big\}\end{aligned} \tag{35}$$

where $H_1(R)$ is the hydrophobicity value, $H_2(R)$ is the hydrophilicity value, and $M(R)$ is the side-chain mass of the amino acid R_i. These quantities were converted from the original hydrophobicity, original hydrophilicity, and original side-chain mass values by standard conversion as follows:

$$H_1(R_i) = \frac{H_1^\circ(R_i) - \dfrac{1}{20}\displaystyle\sum_{k=1}^{20} H_1^\circ(R_k)}{\sqrt{\dfrac{\displaystyle\sum_{y=1}^{20}\left[H_1^\circ(R_y) - \dfrac{1}{20}\displaystyle\sum_{k=1}^{20} H_1^\circ(R_k)\right]^2}{20}}} \tag{36}$$

where $H_1^{\circ}(R_i)$ is the original hydrophobicity value for amino acid R_i that was taken from Tanford [Tan62]; $H_2^{\circ}(R_i)$ and $M^{\circ}(R_i)$ are converted to $H_2(R_i)$ and $M(R_i)$ in the same way. The original hydrophilicity value $H_2^{\circ}(R_i)$ for amino acid R_i was taken from the work of Hopp and Woods [HW81]. The mass $M^{\circ}(R_i)$ of the side chain of amino acid R_i can be obtained from any biochemistry textbook. PseAAC is represented as a vector of size $(20 + \lambda)$ as follows:

$$PseAAC(P_x) = [s_1, ..., s_{20}, s_{21}, ..., s_{20+\lambda}] \qquad x \in (Q, h_1, h_2 \ldots, h_n) \tag{37}$$

where s_i is the pseudo-AAC such that

$$s_i = \begin{cases} \dfrac{f_i}{\sum_{r=1}^{20} f_r + \omega \sum_{j=1}^{\lambda} \theta_j} & 1 \leq i \leq 20 \\ \\ \dfrac{\omega \theta_{i-20}}{\sum_{r=1}^{20} f_r + \omega \sum_{j=1}^{\lambda} \theta_j} & 20 < i \leq 20 + \lambda \end{cases} \tag{38}$$

where f_i is the normalized occurrence frequency of the i^{th} amino acid in the protein sequence, θ_j is the j^{th} sequence-order-correlated factor calculated from Equation 34, and ω is a weight factor for the sequence-order effect. The weight factor ω weights the additional PseAAC components with respect to the conventional AAC components. The user can select any value from 0.05 to 0.7 for the weight factor. The default value given by Chou [Cho01] is 0.05. The mean of individual PseAAC compositions represents the psiPseAAC for Q and was computed as follows:

$$PseAAC_{psi}(Q) = \frac{1}{n+1} \sum PseAAC(P_x) \qquad x \in (Q, h_1, h_2 \ldots, h_n) \tag{39}$$

3.3 SVM

SVMs are powerful machine learning tools used in many biological prediction tools, such as in [MCZ14] and [HPO$^+$19]. We used an SVM with an RBF kernel as implemented in R with the e1071 library (version 1.6-8). The best combination of the C and γ parameters was determined utilizing a grid search approach.

3.4 Annotation transfer by homology

Unlike the discrete representation of a protein sample in the psi-compositions, here, the protein sample was represented by its amino acid sequence and used in a similarity search-based tool (BLAST) to find similar matches in the `TCDB` [SJRT+15]. The `TCDB` uses the classification system approved by the IUBMB for membrane transport proteins, known as the TC system. The `TCDB` is a curated database of accurate and experimentally characterized transporters from over 10,000 published references. If the BLAST search produced a hit, the query was predicted to be a transporter. Since applied thresholds play an essential role in the quality of prediction, different thresholds were utilized, as shown in Table 11.

Table 11: Different BLAST thresholds on the `TCDB`

Name	**BLAST Threshold**	**Motivation**
TCDB_exact	e-value=0; percent identity=100%	exact match
TCDB_high	e-value $\leq 1e-20$; percent identity $\geq$ 40%; query coverage $\geq$ 70%; subject coverage $\geq$ 70%; and difference in length of $\leq$ 10%	thresholds recommended by Butler et al. [AB17] for `TCDB` BLAST
TCDB_med	e-value $\leq 1e-8$	threshold recommended by Barghash et al. [BH13] as an acceptable normalized BLAST threshold when dealing with a TC system

3.5 *TooT-T* methodology

TooT-T solves the binary classification problem in which a membrane protein is classified as a *transporter* or *non-transporter*. Three SVM classifiers were trained with the protein psiAAC, psiPAAC, and psiPseAAC encodings. The prediction from the three SVM models, in addition to the predictions from homology annotation transfer with the three thresholds (TCDB_exact, TCDB_high, and TCDB_med), are combined using an ensemble technique known as stacked generalization or stacking [Wol92]. Instead of combining the predictions from multiple predictors using a simple function (such as voting), stacking trains a new model to aggregate the results.

The stacking framework involves two levels of learning. The first level contains *base classifiers*, which learn directly from the training data. The second level contains a *meta-classifier*, which is trained using the predictions from the base classifiers. The training instances of the meta-classifier were generated while performing cross-validation. Algorithm 1 illustrates how the training dataset of the meta-classifier is generated [Agg14].

Algorithm 1 Stacking with K-fold cross-validation

```
Require: Training data $\mathcal{D} = \{\boldsymbol{x_i}, y_i\}(\boldsymbol{x_i} \in \mathbf{R}^n, y_i \in \{\text{Transporter, Non-transporter}\})$
Ensure: An ensemble classifier $H$
    Step 1: Adopt the cross-validation approach in preparing a training set for the meta-classifier
    Randomly split $\mathcal{D}$ into $K$ equal-sized subsets: $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K\}$
    for $k \leftarrow 1$ to $K$ do
        Step 1.1: learn the base classifiers
        for $t \leftarrow 1$ to $T$ do
            Learn a classifier $h_{kt}$ from $\mathcal{D} \setminus \mathcal{D}_k$
        end for
        Step 1.2: Construct a training set for the meta-classifier
        for $\boldsymbol{x_i} \in \mathcal{D}_k$ do
            obtain $\{\boldsymbol{x'_i}, y_i\}$, where $\boldsymbol{x'_i} = \{h_{k1}(\boldsymbol{x_i}), h_{k2}(\boldsymbol{x_i}), \ldots h_{kT}(\boldsymbol{x_i})\}$
        end for
    end for
    Step 2: Learn the meta-classifier
    Learn a new classifier $h'$ from the collection $\{\boldsymbol{x'_i}, y_i\}$
    Step 3: Relearn the base classifiers using all the data
    for $t \leftarrow 1$ to $T$ do
        Learn a classifier $h_t$ based on $\mathcal{D}$
    end for
    return $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), h_1(\boldsymbol{x}), \ldots, h_T(\boldsymbol{x}))$
```

A GBM, as implemented by the *caret* package in R, was utilized to develop the meta-classifier.

3.6 Performance evaluation

The performances of different models were evaluated on the training dataset using 10-fold cross-validation (10-CV), in which the training dataset was randomly split into ten equally sized sets. A single set was retained as the validation data, and the remaining nine sets were used to train each model. The trained model was then tested using the validation set. The cross-validation process was repeated ten times, where each set was used once as the validation data. The performance evaluation of the 10-CV approach was calculated globally by counting the total true positives, true negatives, false negatives and false positives in all 10 runs (the microaverage).

The cross-validation performance can vary with different random splits; to obtain a more stable error estimation, the 10-CV approach was repeated ten times, with different random splits. The average performance of the 10 runs was calculated, and the variations between the performances were captured by computing the standard deviation (SD). It has been reported that the repeated version stabilizes the error estimation and therefore reduces the variance in the K-CV estimator [Koh95]. Throughout the rest of this chapter, the cross-validation performance is reported as the $means \pm SD$s for the ten different runs of the 10-CV approach.

Furthermore, the independent testing set was also used to perform a thorough evaluation experiment. The data in the independent testing set were not used during the training process and are completely unknown to our models. Four main evaluation metrics were used to evaluate the performance: the sensitivity, specificity, accuracy, and MCC. The sensitivity calculates the proportion of positive samples (transporters) that are correctly identified:

$$Sensitivity = \frac{TP}{TP + FN} \tag{40}$$

The specificity calculates the proportion of non-transporters that are correctly identified:

$$Specificity = \frac{TN}{TN + FP} \tag{41}$$

The accuracy calculates the number of correct predictions divided by the total number of predictions:

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{42}$$

The MCC is less influenced by imbalanced tests because it takes into account true and false positives and negatives. The MCC values range from 1 to -1, where 1 indicates a perfect prediction, 0 represents no better than random, and -1 implies total disagreement between the prediction and observation. Higher MCC values mean that the predictor has high accuracy with positive and negative classes, as well as low misclassification with the two classes. The MCC is considered the best singular assessment metric when the data are imbalanced [Din11] [WP03] [BDA13].

$$MCC = \frac{(TP \times TN - FP \times FN)}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \tag{43}$$

4 Results and discussion

4.1 Performance of the different encodings

The goal is to find the most discriminative encoding to represent a protein sequence; Table 12 presents the cross-validation performance of the SVM models with various encodings. The examined encodings are the baseline compositions where no evolutionary information is incorporated (AAC, PAAC, PseAAC), the PSSM that is commonly used to encode the evolutionary information (implemented as described in [MCZ14] using the same psi-composition thresholds (3 iterations, e-value cutoff of 0.001)), compositions computed from the sequences retrieved from the BLAST search (blast-AAC, blast-PAAC, blast-PseAAC) (e-value cutoff 0.001), and the proposed encodings (psiAAC, psiPAAC, psiPseAAC). Since the training data of the transporter classifier are imbalanced, we focused on the MCC to evaluate the performances of the different models.

The baseline compositions did not show great variations in performance and had an average MCC of 0.65. The MCC was further boosted when the evolutionary information was incorporated. While the PSSM is most commonly applied in the literature to encode evolutionary information, the results suggest that the other encodings that combine AAC with evolutionary information yield a higher accuracy. Since the PSSM encoding is extracted from the PSI-BLAST output, we expected it to show an improved performance compared to at least the BLAST compositions, but this phenomenon was not what was illustrated by our results. One explanation for this finding could be that the commonly used PSSM encoding is computed from the original PSSM profile output to make it fixed at size 20×20, and this PSSM encoding, although superior to the baseline, does not capture properties to the extent shown by the amino acid compositions. Among all of the tested encodings, psiPAAC achieved the highest MCC of approximately 0.90. The use of the psiPAAC encoding also achieved promising results in the detection of other membrane functional classes, such as enzymes and receptors. The experimental details on the psi-composition performance in the other functional classes are available in Appendix B.

The high performance achieved by the psi-composition encodings is a result of incorporating two distinctive approaches, namely, the amino acid composition and evolutionary information. The idea is that multiple homologous sequences can reveal more about the function of a protein than a single sequence. Homologous sequences can be inferred when they share more similarity than would be expected by chance [Pea13]. Similarity tools such as BLAST help to minimize the number of false positives (non-homologs with significant scores; type I errors) but do not necessarily detect remote homologs (homologs with non-significant scores; type II errors) [Pea13]. PSI-BLAST is more sensitive than BLAST in terms of finding such remote homologs and is thus utilized by the proposed encodings. Furthermore, the alignment results of PSI-BLAST contain valuable information about the most conserved regions in the protein, and such conservation can reflect the function of the protein. Computing the average amino acid composition from the aligned homologous sequences thus provides a better indication of the function with less noise than computing the composition from a single sequence.

The impact of incorporating different sources of evolutionary information is presented in Table 13. The compositions computed from a single BLAST search had an average improvement over baseline of 37.83%. The psi-composition further enhanced the baseline MCC by 39.94%. The improved performance between psi-compositions and BLAST compositions was expected because, unlike BLAST, which uses only a general scoring matrix, PSI-BLAST uses a PSSM to detect sequences with a similar conservation pattern to the PSSM, thus making PSI-BLAST more sensitive to weak but biologically significant sequence relationships than BLAST [AMS$^+$97].

Table 12: Transporter detection performances of the different models

Encoding	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
psiPAAC*	88.45 ± 0.17	98.77 ± 0.05	96.17 ± 0.04	0.8970 ± 0.0012
psiAAC *	87.83 ± 0.23	98.66 ± 0.06	95.93 ± 0.07	0.8905 ± 0.0019
blastPAAC	87.50 ± 0.22	98.60 ± 0.06	95.80 ± 0.05	0.8869 ± 0.0015
blastPseAAC	87.31 ± 0.32	98.53 ± 0.07	95.70 ± 0.11	0.8800 ± 0.003
psiPseAAC*	87.10 ± 0.25	98.28 ± 0.08	95.47 ± 0.08	0.8800 ± 0.0021
blastAAC	85.21 ± 0.24	98.12 ± 0.08	94.87 ± 0.08	0.8613 ± 0.0023
PSSM	80.16 ± 0.23	97.17 ± 0.10	92.88 ± 0.10	0.8063 ± 0.0027
PAAC	65.01 ± 0.21	95.81 ± 0.10	88.05 ± 0.08	0.6662 ± 0.0024
AAC	59.78 ± 0.37	95.67 ± 0.15	86.62 ± 0.14	0.6225 ± 0.0041
PseAAC	59.91 ± 0.27	95.40 ± 0.09	86.45 ± 0.12	0.6179 ± 0.0034

This table shows the *means* ± *SDs* of the ten different 10-CV runs, in ascending order of the MCC. The asterisk (*) refers to the models used in *TooT-T* .

Table 13: Impact of various factors on performance.

Encoding	**MCC**			**blastX to X**		**psiX to X**		**psiX to blastX**	
X	X	blastX	psiX	Delta	Percent	Delta	Percent	Delta	Percent
AAC	0.62	0.86	0.89	0.240	38.71	0.270	43.55	0.030	3.49
PAAC	0.67	0.89	0.90	0.220	32.84	0.230	34.33	0.010	1.12
PseAAC	0.62	0.88	0.88	0.260	41.94	0.260	41.94	0.000	0.00
Average				0.24	37.83	0.25	39.94	0.01	1.54

This table notes the difference in the MCC, delta, and percentage improvement in the MCC, when incorporating different evolutionary information with the baseline compositions. The highest improvement in the accuracy was achieved by the psi-compositions, with an average improvement of 39.94% compared to the baseline.

4.2 Performance of annotation transfer by homology

The performance of annotation transfer by homology to detect transporters against the TCDB under different thresholds is presented in Table 14. The choice of proper similarity thresholds is critical, as shown in Table 14, and there is a trade-off between sensitivity and specificity, where stricter thresholds (TCDB_exact) result in a low true transporter detection rate (sensitivity) but more reliable elimination of non-transporters (specificity).

However, when the thresholds are set to be more tolerant (TCDB_med), the percentage of transporters detected increases but at the cost of more false predictions. A good balance between sensitivity and specificity was achieved using the TCDB_high thresholds, where the overall MCC reached 0.68, which is lower than the best machine learning method psiPAAC with an MCC of 0.89. Nevertheless, this gives a different viewpoint, which we utilize in the *TooT-T* ensemble classifier.

Table 14: Performance of annotation transfer by homology

Threshold	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
TCDB_exact	47.62	99.80	86.72	0.6329
TCDB_high	81.56	89.52	87.52	0.6844
TCDB_med	95.87	60.08	69.10	0.4873

This table shows the performance of homology annotation transfer with the training dataset using different thresholds. The best prediction power was achieved using the TCDB_high threshold. The predicted transporter from TCDB_exact was more reliable than that from the other thresholds due to the high specificity.

4.3 *TooT-T* ensemble performance

The performance of the ensemble classifier and each of its constituent classifiers in the cross-validation and independent testing set is presented in Table 15 and in Table 16, respectively. The ensemble classifier consistently outperformed its constituent classifiers in detecting transporters (sensitivity) while maintaining a credible true negative rate. Overall, it surpassed all of the other models in terms of the accuracy and MCC.

It was previously shown by [OS96] and [KV95] that ensemble classifiers benefited the most when the individual classifiers comprising the ensemble were both *accurate* and had *low correlation* (i.e., making errors in different parts of the input space). The constituent classifiers in our ensemble achieved the highest MCCs, and the correlations between them are presented in Table 17. When combining the predictions of only the three models on the machine learning side, we observed no improvement in the overall accuracy. This finding is reasonable, since the predictions from the machine learning models in our case were highly correlated. The obtained performance was mainly achieved by combining a different view, annotation transfer by homology, which has lower correlation than SVM-based classifiers.

Table 15: Cross-validation performance of the *TooT-T* model

	Name	Sensitivity	Specificity	Accuracy	MCC
SVM	psiAAC	87.83 ± 0.23	98.66 ± 0.06	95.93 ± 0.07	0.8905 ± 0.0019
	psiPAAC	88.45 ± 0.17	98.77 ± 0.05	96.17 ± 0.04	0.8970 ± 0.0012
	psiPseAAC	87.10 ± 0.25	98.28 ± 0.08	95.47 ± 0.08	0.8800 ± 0.0021
ATH	TCDB_exact	47.62	99.80	86.72	0.6329
	TCDB_high	81.56	89.52	87.52	0.6844
	TCDB_med	95.87	60.08	69.10	0.4873
	TooT-T	**91.80 ± 0.08**	**98.79 ± 0.01**	**97.02 ± 0.02**	**0.9203 ± 0.000**

This table lists the *means* ± *SD*s of the ten different runs of the 10-CV of the proposed ensemble. It also shows the performance of each of its constituent classifiers. ATH: annotation transfer by homology

Table 16: Independent testing performance of the proposed model

	Name	Sensitivity	Specificity	Accuracy	MCC
SVM	psiAAC	88.74	98.18	95.81	0.8872
	psiPAAC	89.64	98.49	96.26	0.8995
	psiPseAAC	87.84	98.18	95.58	0.8809
ATH	TCDB_exact	38.74	100	84.6	0.5668
	TCDB_high	80.18	88.50	86.41	0.6582
	TCDB_med	96.4	59.61	68.86	0.4879
	TooT-T	**93.69**	**98.49**	**97.28**	**0.9274**

This table shows the performance of the proposed ensemble and each of its constituent classifiers trained on *DS-T* training set and tested on *DS-T* independent testing set. ATH: annotation transfer by homology

Table 17: Phi correlation coefficients of the constituent classifiers

model	**psiAAC**	**psiPAAC**	**psiPseAAC**	**TCDB_exact**	**TCDB_high**	**TCDB_med**
psiAAC	1.00	0.97	0.96	0.61	0.64	0.47
psiPAAC	0.97	1.00	0.95	0.61	0.64	0.47
psiPseAAC	0.96	0.95	1.00	0.60	0.64	0.47
TCDB_exact	0.61	0.61	0.60	1.00	0.59	0.35
TCDB_high	0.64	0.64	0.64	0.59	1.00	0.58
TCDB_med	0.47	0.47	0.47	0.35	0.58	1.00

This table shows the correlation between the constituent classifiers of the ensemble. Among them, the homology annotation transfer exhibited a lower correlation than the machine learning models. This lower correlation motivates the use of ensemble techniques and helps to build a more powerful model.

4.4 Comparative performance

The experiments suggest that the *TooT-T* methodology is effective in detecting transporters. Since most of the state-of-the-art tools are not accessible and their source codes are not available, we performed *TooT-T* methodology on the same TrSSP dataset [MCZ14] used to train and test the other state-of-the-art transporter predictors to properly compare them to the *TooT-T* method.

Table 18 compares the performance of the *TooT-T* method with the methods from other published works on the same dataset. The highest prediction accuracy was achieved by the method proposed by Li et al. [LLX+16]. The high performance achieved by their model was mainly due to the use of the protein `GO` annotations as features. Such a high performance is to be expected, considering the fact that all of the sequences in the benchmark dataset were well annotated and extracted from the `Swiss-Prot` database. The goal of *TooT-T* is to predict *novel* and *unannotated* transporter proteins.

Table 18: Comparison with methods from other published works

Tool	**Sensitivity**		**Specificity**		**Accuracy**		**MCC**	
	Ind.	CV	Ind.	CV	Ind.	CV	Ind.	CV
SCMMTP [LVY+15]	80.00	83.76	68.33	77.68	76.11	81.12	0.47	0.62
TrSSP [MCZ14]	76.67	76.67	81.67	78.46	80.00	78.99	0.57	0.58
Ho et al. [HPO+19]	100.00	83.14	77.50	84.48	85.00	83.94	0.73	0.68
TooT-T	**94.17**	**90.15**	**88.33**	**89.97**	**92.22**	**90.07**	**0.82**	**0.80**
Li et al. [LLX+16]	96.67	99.50	95.83	97.44	96.11	98.33	0.91	0.97

The other methods did not incorporate annotations of proteins as features and relied solely on the protein sequence to extract features to distinguish between transporters and non-transporters. They therefore provide a better comparison for the proposed method. The method proposed by Ho et al. [HPO+19] achieved a better sensitivity (100%) than *TooT-T* (94.17%) in the independent dataset. However, the specificity was 77.50% compared to the 88.33% achieved by *TooT-T*. The proposed method achieved a higher accuracy (↑ 7%) and a higher MCC (↑ 0.09) than the method proposed by Ho et al. [HPO+19] in transporter detection. Overall, *TooT-T* achieved a better accuracy, specificity, and MCC than all of the tools reported in published works, both in independent and cross-validation testing.

5 Conclusion

We propose the *TooT-T* ensemble classifier, which can distinguish transporter membrane proteins from other proteins. The ensemble classifier is trained to optimally combine the predictions obtained from machine learning and homology annotation methods to produce a final prediction. The machine learning components of the ensemble consist of SVM models that incorporate a novel encoding method, *psi-composition*. The psi-composition method combines traditional AAC with the alignment results of PSI-BLAST and shows superior prediction performance to models built using other features, including the PSSM profile. While the predictions obtained from annotation transfer by homology were not superior to the best machine learning models, they provided a different viewpoint of the solution. The proposed ensemble exploits the fact that different methods misclassify different sequences to build a more credible model. It was demonstrated through repeated 10-CV and independent dataset tests that the proposed ensemble outperformed its constituent classifiers and all other state-of-the-art transporter predictors that rely on the protein sequence alone.

Chapter 6

Ontology-based transporter substrate annotation for benchmark datasets

This chapter addresses the third research objective:

O3: To facilitate the data collection process in a traceable and reproducible manner

Some parts of this chapter have been presented and published at the *BIBM* conference: Alballa, M., & Butler, G. (2019, November). Ontology-based transporter substrate annotation for benchmark datasets. In 2019, *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (pp. 2613-2619).

This chapter is organized as follows: Section 1 introduces the chapter. Section 2 delineates the challenges faced when building a substrate-specific transport protein dataset and highlights the inconsistencies among the gold standard databases. Section 3 elucidates the proposed ontology-based tool, *Ontoclass*. Section 4 presents two case studies; the first case study (Section 4.1) compares *Ontoclass* annotation with a manually curated dataset, and the second case study (Section 4.2) reflects the number of annotated transporters and their substrates in the `Swiss-Prot` database. Section 5 discusses the findings, and Section 6 concludes the chapter.

1 Introduction

Membrane transport proteins perform a fundamental biological task of controlling the influx of essential nutrients and ions into the cell and the efflux of cellular waste and toxins out of the cell [G$^+$06]. The identification of the substrate specificity of a transporter is essential to understanding its function and to develop drugs [G$^+$06]. Predicting the substrate specificities of transporters has been the focus of many studies [COLG11, SH12, BH13, MCZ14, HPO$^+$19].

In the context of transporters and transported substrates, the ultimate objective is to predict the exact transported substrate (e.g., arginine). However, the limited number of annotated substrates makes the prediction possible only at a high level of abstraction (e.g., amino acids), where all of the transporters that transport a group of substrates are combined together in one class.

Data collection is the backbone of any research; constructing a substrate-specific transport protein dataset for substrate prediction generally follows a manual curation process, in which a class is assigned to sequences that transport substrates with similar chemical properties. Unlike the manual curation of major biological databases such as `Swiss-Prot`, where the manual curation process is well defined and the entries are handled in a consistent manner, the manual curation of transporter substrate benchmark datasets is generally imprecise; details behind the class assignment process are rarely described, which makes reproducing the same grouping very difficult.

The goal of this work is to define a reliable method to automate the data collection process and make establishing, updating or expanding a transporter substrate dataset achievable for any user. To automate the data collection process, two main components are needed: a source from which the transporter substrate annotation is found and a consistent method to assign broader classes to more specific substrates (e.g., amino acids to arginine). We found that the `GO` MF annotation in the `Swiss-Prot` database contains information about transported substrates that are carefully curated by the database curators, and the well-structured `ChEBI` ontology for biochemical entities has already established the relationships between chemical entities. This work therefore proposes an automated tool, *Ontoclass*, that can assign a substrate class to a transporter in a reliable and consistent manner without an external investigator's input. *Ontoclass* takes `UniProt` identifiers as input and utilizes the `GO` annotation from `Swiss-Prot` to detect transporters and assign their substrate specificities. Then, the tool utilizes the `ChEBI` ontology to determine the substrate class. The tool outputs the assigned substrate classes and related information.

2 Challenges

2.1 Different transporter classifications

There are two main gold standard databases for transporters: the `TCDB` and the `Swiss-Prot` database. `TCDB` is a curated database of accurate and experimentally characterized information collected from over 10,000 published references; as of March 2020, `TCDB` contains over 19,500 entries, which are classified into 1,448 transporter families. `TCDB` uses a hierarchical classification system approved by the IUBMB for membrane transport proteins; it is known as the TC system. The `Swiss-Prot` database is the primary worldwide database of well-annotated and manually inspected data; as of March 2020, `Swiss-Prot` contains 561,611 entries. The `Swiss-Prot` database adopts the `GO` terms for its curation, and more than 28,103 proteins are characterized with the `GO` MF term `GO:0005215 transporter activity`).

Proteins classified in one database are not necessarily included in another classification. For example, only 37% (7,223 of the 19,500) of the `TCDB` annotated transporters are also annotated in the `Swiss-Prot` database. Of those transporters, only 3,618 were annotated with the transporter-related `GO` MF term `GO:0005215 transporter activity`. Such inconsistencies complicate the process of finding transporters and annotating them with a substrate class.

2.2 Lack of documentation for manual class assignment

Several datasets have been proposed and used to predict the substrate specificity of transporters; all of them use manual curation to assign substrate classes to transporters. In 2011, Chen et al. [COLG11] defined four substrate classes: *electrons*, *proteins/mRNAs*, *ions*, and *others*. Their dataset is not tailored to a specific organism and contains a total of 651 transporters. In 2012, Schaadt et al. [SH12] produced an *Arabidopsis thaliana* dataset with a total of 61 transporter proteins that belong to four substrate classes: *amino acids*, *oligopeptides*, *phosphates*, and *hexoses*. In 2013, Barghash et al. [BH13] considered four substrate classes, *metal ions*, *phosphates*, *sugars*, and *amino acids* transporters from *Escherichia coli* (72 transporters), *Saccharomyces cerevisiae* (79 transporters), and *Arabidopsis thaliana* (95 transporters). In 2014, the goal of a study by Mishra et al. [MCZ14] was to classify transport proteins into the maximum possible number of classes according to their transported substrates. To achieve this goal, a substrate-specific transport protein dataset with a total of 900 transporters was constructed. The dataset consisted of seven transporter classes: *amino acids/oligopeptides*, *anions*, *cations*, *electrons*, *proteins/mRNAs*, *sugars*, and *others*. In 2019, Ho et al. [HPO$^+$19] created a new dataset that contained 1,197 transporters that belong to seven classes: *amino acids*, *electrons*, *hydrogen ions*, *lipids*, *proteins/mRNAs*, *sugars*, and *others*.

As illustrated above, there is no specific set of transporter classes used in all of the datasets. Some authors (Chen et al. [COLG11]) group the substrates into four groups with one general class, *others* , referring to all other substrate types. Other authors (Schaadt et al. in [SCH10] and [SH12]) include oligopeptides (i.e., a few amino acids linked in a polypeptide chain). Other studies (Chen et al. [COLG11] and Mishra et al. [MCZ14]) elect to incorporate proteins/mRNAs, which consist of one or more polypeptides with at least 50 amino acids. Conversely, others (Barghash et al. [BH13]) completely ignore the protein or oligopeptide category. Additionally, the boundaries based on which the substrate is assigned to a class are not clear, which makes expanding the dataset to accumulate new proteins exceptionally challenging. For example, should the transporter protein with `UniProt`-ID `Q10901` that transports *L-glutamate*, an α-amino acid anion, be assigned to the *amino acids* class, the *anions* class, or both?

3 The proposed tool: *Ontoclass*

3.1 Overview

The tool takes `UniProt` identifiers of proteins as input. For each protein, it determines whether the protein has transporter-related `GO` MF annotations in the `Swiss-Prot` database. If a transporter-related `GO` MF is found, it looks for the `ChEBI` identifier of the transported substrates in the `GO` annotation. This `ChEBI`-ID and its ancestors in the `ChEBI` ontology are used to find the most specific substrate class according to a predetermined list of substrate classes and `ChEBI`-IDs. The tool outputs the final substrate class of each protein along with additional information (Section 3.5). Details regarding the steps are presented in the following subsections.

3.2 Substrate classes to **`ChEBI`**-ID

To assign a substrate class to a transporter protein, it is necessary to determine the substrate classes that the tool produces. The first decision we had to make was to choose the substrate categories with respect to the `ChEBI`-IDs. We initially attempted to follow Saier's classification system [Sai00] (see Table 19) by mapping each subcategory to its relevant `ChEBI` term, but we encountered multiple issues.

First, the classification system simultaneously offers role and chemical classifications. For example, category five (vitamins, cofactors, and their precursors) and some of the subcategories of category six belong to `ChEBI`'s role ontology, while the rest of the categories belong to its chemical ontology. A single compound could have both if it includes the *"is a"* and *"has a role"* relationships in its ontology. Therefore, all of the compounds have chemical classifications, and some also have role classifications. We would often encounter the issue of multiple classifications for a single substrate. For example, *glycine* (`CHEBI:15428`), which is an amino acid, is classified under Saier's classification as follows:

```
3.A Amino acids and conjugates; 5.D Signaling molecules;
6.B Specific drugs
```

Since we are mostly interested in the chemical composition of the transported substrates, and because of all of the substrate prediction methods used to predict the chemical classification, we opted to consider the chemical categories.

The second issue concerns the fact that the groupings from Saier's classification system categories and the ChEBI ontology are not consistent. For example, there is no corresponding ChEBI term that corresponds to the *2.A Sugar polyols and their derivatives* subcategory but rather two different terms, *polyols* (CHEBI:26191) and *monosaccharides* (CHEBI:63367). The closest common ancestor between these two terms is *organic molecular entities* (CHEBI:50860). Similarly, *monosaccharides* (CHEBI:63367) and *carbohydrates* (CHEBI:16646) share the ancestor *carbohydrates and carbohydrate derivatives* (CHEBI:78616) in the ChEBI ontology but are not in the same major category in Saier's classification system.

Since we rely on the ChEBI ontology in our automatic substrate assignment scheme, we modified Saier's classification system to be consistent with the ChEBI ontology. Figure 19 depicts categories in Saier's classification system with respect to the ChEBI ontology, where the edges represent *"is a"* relationships. Table 20 groups the categories according the relevant closest ancestor in agreement with the ChEBI ontology. We call this mapping *S2C*.

Table 19: Saier's classification of transporter substrates [Sai00]

Category and substrate type	**Subcategories**
1. Inorganic molecules	A. Nonselective B. Water C. Cations D. Anions E. Others
2. Carbon compounds	A. Sugars, polyols, and their derivatives B. Monocarboxylates C. Di- and tricarboxylates D. Noncarboxylate organic anions (organophosphates, phosphonates, sulfonates, and sulfates) E. Others
3. Amino acids and their derivatives	A. Amino acids and conjugates B. Amines, amides, and polyamines C. Peptides D. Other related organocations E. Others
4. Bases and their derivatives	A. (Nucleo)bases B. Nucleosides C. Nucleotides D. Other nucleobase derivatives E. Others
5. Vitamins, cofactors, and their precursors	A. Vitamins and vitamin or cofactor precursors B. Enzyme and redox cofactors C. Siderophores; siderophore-Fe complexes D. Signaling molecules E. Others
6. Drugs, dyes, sterols, and toxics	A. Multiple drugs B. Specific drugs C. Bile salts and conjugates D. Sterols and conjugates
7. Macromolecules	A. Carbohydrates B. Proteins C. Nucleic acids D. Lipids E. Others
8. Miscellaneous compounds	

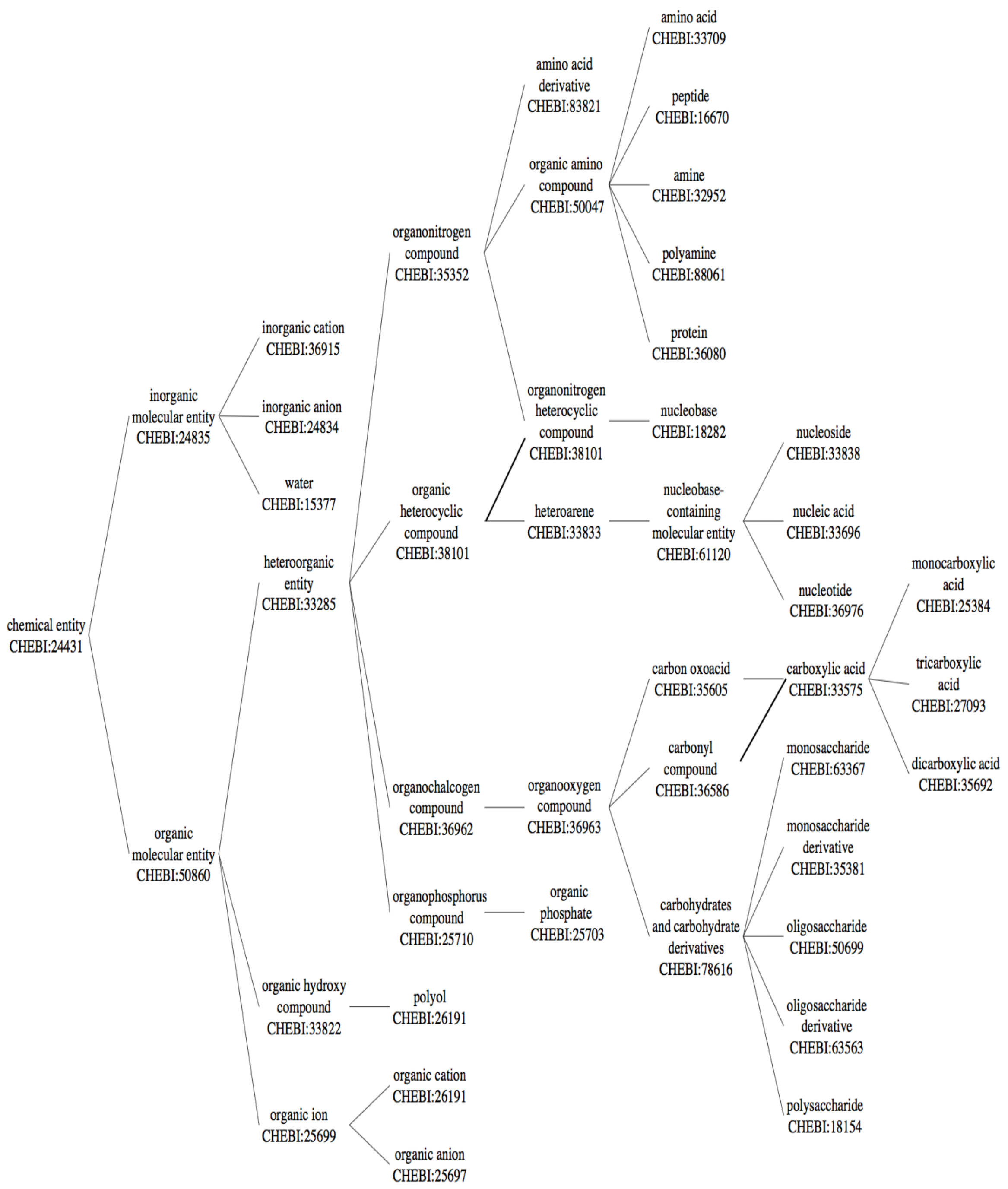

chemical entity CHEBI:24431
inorganic molecular entity CHEBI:24835
organic molecular entity CHEBI:50860
inorganic cation CHEBI:36915
inorganic anion CHEBI:24834
water CHEBI:15377
heteroorganic entity CHEBI:33285
organic hydroxy compound CHEBI:33822
organic ion CHEBI:25699
organonitrogen compound CHEBI:35352
organic heterocyclic compound CHEBI:38101
organochalcogen compound CHEBI:36962
organophosphorus compound CHEBI:25710
polyol CHEBI:26191
organic cation CHEBI:26191
organic anion CHEBI:25697
amino acid derivative CHEBI:83821
organic amino compound CHEBI:50047
organonitrogen heterocyclic compound CHEBI:38101
heteroarene CHEBI:33833
organooxygen compound CHEBI:36963
organic phosphate CHEBI:25703
amino acid CHEBI:33709
peptide CHEBI:16670
amine CHEBI:32952
polyamine CHEBI:88061
protein CHEBI:36080
nucleobase CHEBI:18282
nucleobase-containing molecular entity CHEBI:61120
carbon oxoacid CHEBI:35605
carbonyl compound CHEBI:36586
carbohydrates and carbohydrate derivatives CHEBI:78616
nucleoside CHEBI:33838
nucleic acid CHEBI:33696
nucleotide CHEBI:36976
carboxylic acid CHEBI:33575
monosaccharide CHEBI:63367
monosaccharide derivative CHEBI:35381
oligosaccharide CHEBI:50699
oligosaccharide derivative CHEBI:63563
polysaccharide CHEBI:18154
monocarboxylic acid CHEBI:25384
tricarboxylic acid CHEBI:27093
dicarboxylic acid CHEBI:35692

Figure 19: Simplified view of `ChEBI` ontology terms

This figure shows a simplified view of the categories in Saier's classification system with respect to the `ChEBI` ontology; the edges represent *"is a"* relationships in the `ChEBI` ontology; some edges were omitted to simplify the view. Each node contains the `ChEBI` term and the relevant `ChEBI`-ID. The leaves
are the categories.

Table 20: Classification of transport system substrates using the ChEBI ontology

Category and type of substrate	**Class**	**ChEBI-ID**
1. Inorganic molecules	A. Nonselective	CHEBI:36914 CHEBI:24431
	B. Water	CHEBI:15377
	C. Cations	CHEBI:36915
	D. Anions	CHEBI:24834
2. Organic ions	A. Organic cations	CHEBI:25697
	B. Organic anions	CHEBI:25696
3. Carbohydrates and derivatives	A. Monosaccharides and derivatives	CHEBI:35381 CHEBI:63367
	B. Oligosaccharides and derivatives	CHEBI:50699 CHEBI:63563
	C. Polysaccharides and derivatives	CHEBI:18154 CHEBI:65212
4. Carboxylic acids	A. Monocarboxylic acids	CHEBI:25384
	B. Tricarboxylic acids	CHEBI:27093
	C. Dicarboxylic acids	CHEBI:35692
5. Organonitrogen compounds	A. Amino acids	CHEBI:33709
	B. Amino acid derivatives	CHEBI:83821
	C. Peptides	CHEBI:16670
	D. Amines	CHEBI:32952
	E. Polyamines	CHEBI:88061
	F. Proteins	CHEBI:36080
	G. Other organic amino compounds	CHEBI:50047
6. Organic heterocyclic compounds	A. Nucleobases	CHEBI:18282
	B. Nucleosides	CHEBI:33838
	C. Nucleic acids	CHEBI:33696
	D. Nucleotides	CHEBI:36976
7. Miscellaneous	A. Polyols	CHEBI:26191
	B. Organic phosphates	CHEBI:25703
	C. Amides	CHEBI:32988
	D. Other organic molecular entities	CHEBI:50860

This table corresponds to a map from the substrate class to the ChEBI-ID (*S2C*). The first two columns represent a modified version of Saier's classification system [Sai00], the last column shows the corresponding ChEBI-IDs.

3.3 Mapping GO terms to substrate classes

This stage constructs a lookup table, *GO2C*, which maps transporter-related terms, i.e., descendants of the GO MF, GO:0005215 transporter activity, to the ChEBI-ID of the most specific substrate class. Algorithm 2 constructs the lookup table *GO2C*. First, all descendants of the GO MF term (GO:0005215 transporter activity) and their corresponding transported substrate ChEBI-ID mappings were obtained from the *go-plus.owl* ontology file downloaded from http://snapshot.geneontology.org/ontology/extensions/go-plus.owl Then, related classes in Table 20 that are in, or ancestors of, the ChEBI-ID in the ChEBI ontology, available at ftp://ftp.ebi.ac.uk/pub/databases/chebi/ontology/chebi_lite.obo, were mapped to that term. This initial mapping was further filtered to retain only the most specific substrate class. For example, the initial mapping could represent 1.A (*Nonselective*), 5.A (*amino acids*), 5.G (*another organic amino compounds*), 7.D (*other organic*), and the concise class is 5.A (*amino acids*). Table 21 presents samples of the *GO2C* lookup table.

Algorithm 2 Construction of the GO2C mappings: This algorithm constructs a *map* from the GO MF term descendants of the GO MF (GO:0005215 transporter activity) → ChEBI-ID

```
Require: GO as Gene Ontology
Require: ChEBI as ChEBI ontology
Require: $S2C$ mapping
Ensure: $GO2C$ mapping
  Initiate $GO2C$ as an empty map
  for term $t \in$ descendants from GO:0005215 do
    $tc \leftarrow$ ChEBI term in t
    $tcs \leftarrow$ most specific ChEBI term in $ancestors(tc) \cap range(S2C)$
    add item $[t \rightarrow tcs]$ to map $GO2C$
  end for
  return $GO2C$
```

Table 21: Samples of the GO2C lookup table

`GO` MF term	`ChEBI`-ID	Substrate class
`GO:0005249 voltage-gated potassium channel activity`	`CHEBI:29103`	1.C Cations
`GO:0015105 arsenite transmembrane transporter activity`	`CHEBI:29866`	1.D Anions
`GO:0005277 acetylcholine transmembrane transporter activity`	`CHEBI:15355`	2.B Organic cations
`GO:0015136 sialic acid transmembrane transporter activity`	`CHEBI:26667`	3.A Monosaccharides and derivatives
`GO:0071913 citrate secondary active transmembrane transporter activity`	`CHEBI:30769`	4.B Dicarboxylic acids
`GO:0015193 L-proline transmembrane transporter activity`	`CHEBI:17203`	5.A Amino acids
`GO:0015638 microcin transmembrane transporter activity`	`CHEBI:64627`	5.C Peptides
`GO:0005340 nucleotide-sulfate transmembrane transporter activity`	`CHEBI:64702`	6.D Nucleotides
`GO:0015255 propanediol channel activity`	`CHEBI:26288`	7.A Polyol
`GO:0031927 pyridoxamine transmembrane transporter activity`	`CHEBI:16410`	7.D Other organic molecular entities

This table shows example entries in the lookup table. Each entry contains the `GO` MF term, the `ChEBI`-ID of the transported substrate, and the general class that the substrate belongs to as defined by Table 20.

3.4 Class assignment with confidence

Given the `UniProt`-ID of a protein, the `GO` MF annotations of that protein from the `Swiss-Prot` database are examined. Each `GO` MF annotation in the *GO2C* lookup table conveys that the protein is a transporter with the same transporter substrate class in the lookup table. If the protein contains multiple `GO` MF terms in the *GO2C* lookup table, then all of the corresponding classes are assigned to that protein. The assigned classes are then filtered to retain only the most specific class, as presented in Algorithm 3. Furthermore, the evidence of each class assignment is provided, as determined by the GO annotation in Swiss-Prot for that sequence, which provides an impression of the confidence level of the assignment of that substrate.

Algorithm 3 Construction of the U2S mapping: This algorithm constructs a *map* from the UniProt-ID of transporters → $\mathbb{P}$(substrate classes)

```
Require: $S2C$ map
Require: $GO2C$ map
Require: identifier $u$ in UniProt of a transporter
Ensure: $s\_set$ is a set of most specific substrate classes of $u$ in the modified Saier's list of substrate classes
    $s\_set \leftarrow$ empty set
    for term $t \in u$ that is descendant of GO:0005215 do
        $s\_set \leftarrow s\_set \cup S2C^{-1}(GO2C(t))$
    end for
    $s\_set \leftarrow s\_set - \{t' \in s\_set \mid \exists\, t \in s\_set$ s.t $t$ is more specific than $t'\}$
    return $s\_set$
```

3.5 Presenting the output

The tool outputs the automatically assigned substrate class of a given protein along with other additional information collected from different sources as presented in Table 22. Such information gives the user a broader view into the query protein and its updates. If the protein does not have any transporter-related GO MF term, the tool will simply output "NA" for that protein.

Table 22: *Ontoclass* substrate assignment output

Name	**Description**
UniProt-ID	Input UniProt-ID of the protein sequence
Auto.Assignment	Ontology-based automated substrate class assignment
Auto.Confidence	Evidence of the annotation
Date	Last modification date of the entry in the Swiss-Prot database
KW	List of keywords as in the Swiss-Prot database
GO.annotation	GO MF annotation of the protein
ChEBI.annotation	ChEBI-ID associated with the transporter-related GO annotation
In.TCDB.	Cross-reference of this protein with the TCDB. The entry could have 3 possible letters Y, H, or N; Y indicates that the protein has an exact match with a TCDB entry, H indicates that there is a hit by homology, and N indicates that there is no hit
TCDB.fam	Corresponding TCDB family of the protein, if there is a hit in TCDB
TCDB.substrate	Substrate annotation in the TCDB for the protein hit.

This table shows the output of the automated tool. In addition to the substrate class mapping, it includes other information from different sources to give the user a general overview.

4 Case studies

4.1 Comparison with manually curated datasets

To assess how well *Ontoclass* establishes different classes relative to other manually annotated datasets, we used the same sequences in Mishra et al.'s TrSSP dataset [MCZ14] and compared the output classes with their manually assigned classes. Since *Ontoclass* includes more substrate classes than the dataset from [MCZ14], we grouped some classes in our output to more general classes in Mishra et al.'s dataset as follows: *organic, inorganic cations* into *cations*; *organic, inorganic anions* into *anions*; *amino acids, amino acid derivatives, peptides* into *amino acids*; *monosaccharides, oligosaccharides* into *sugars.*

Table 23 compares the obtained classes. "No mapping" refers to the sequences that do not have sufficient annotation for our tool to infer the substrate. Those sequences either have a `GO` MF annotation that is a descendant of transporter activity and no substrate mapping (e.g., `GO:0015297 antiporter activity`) or they are sequences that do not have any descendant of transporter activity, such as nearly all of the proteins in Mishra et al.'s *electron* class with the annotation (`GO:0009055 electron transfer activity`), as presented in Figure 20. There are 641 sequences with sufficient `GO` annotations for our tool to infer the substrate class. Of them, 576 (90%) agree with Mishra et al.'s classes, while the other 65 disagree. Several examples that illustrate the disagreement are presented in Table 24.

Table 23: Comparison between the TrSSP dataset and the *Ontoclass* dataset

		Ontoclass		
Transporter class	**TrSSP dataset**	**No mapping**	**Agreement**	**Disagreement**
Amino acids	85	6	73	6
Anions	72	7	57	8
Electrons	70	67	0	3
Cations	296	36	244	16
Proteins	85	57	20	8
Sugars	72	14	54	4
Others	220	72	128	20
Total	900	259	576	65

This table compares the transporter mapping from the TrSSP dataset in Mishra et al. [MCZ14] and those generated by our automated tool. Because the transporter classes are not the same between the two datasets, we mapped our classes to the TrSSP dataset classes as follows: amino acids and amino acid derivatives (5.A, 5.B) into the *amino acids* class; organic and inorganic anions (1.D, 2.B) into the *anions* transporter class; organic and inorganic cations (1.C, 2.A) into the *cations* class; monosaccharides, oligosaccharides and derivatives (3.A, 3.B) into the *sugars* class; and proteins (5.E) to the *proteins* class. The rest of our classes were mapped to *others*. The column labeled **Agreement** indicates that the ontology-based assignment dataset class is the same as the TrSSP dataset class. **No mapping** indicates that there is no corresponding class. **Disagreement** indicates that the ontology-based mapping and the TrSSP dataset have different substrate classes.

Table 24: Examples of disagreement

`UniProt-ID`	`GO` annotations	*Ontoclass* assignment	TrSSP dataset
P46133	GO:0015558; F: secondary active p-aminobenzoyl-glutamate transmembrane transporter activity	2.A Organic anions	Amino
Q12482	GO:0015183; F: L-aspartate transmembrane transporter activity GO:0005313; F: L-glutamate transmembrane transporter activity	2.A Organic anions	Amino
Q20106	GO:0015232; F: heme transporter activity	7.D Other organic entities	Cation
P33941	GO:1904680; F: peptide transmembrane transporter activity GO:0042626; F: ATPase-coupled transmembrane transporter activity	5.C Peptides	Protein
Q10185	GO:0044604; F: ATPase-coupled phytochelatin transmembrane transporter activity GO:0042626; F: ATPase-coupled transmembrane transporter activity	5.C Peptides	Other

This table shows examples where the substrate classes produced by the automated tool and those in the dataset from Mishra et al. [MCZ14] conflict. Only the transporter-related `GO` MF annotations that have `ChEBI` mappings are shown.

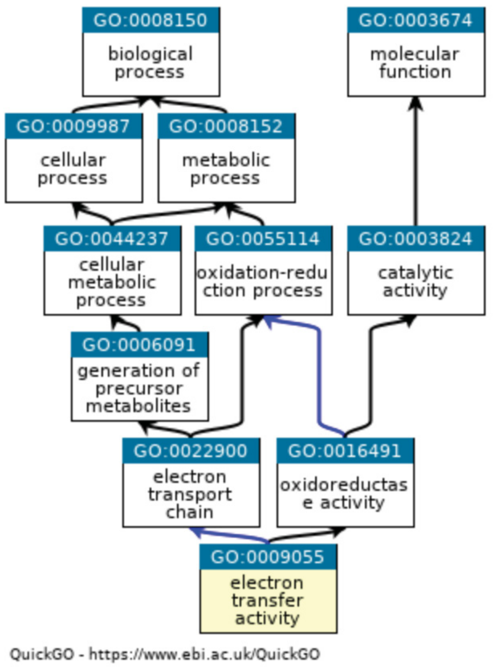


Figure 20: Ancestor chart for `GO:0009055`.
This figure illustrates the ancestor chart for the `GO` MF term `GO:0009055 electron transfer activity`; black edges represent *"is a"* relationships; blue edges represent *"part of"* relationships. This term is not a descendant of a transporter activity term, and it is therefore not recognized as a transporter by *Ontoclass*.

4.2 Building a new dataset

To ascertain the number of available sequences in each substrate class, we extracted all of the sequences from the `Swiss-Prot` database that are located in the membrane and have `GO:0005215 transporter activity` annotations. The data were then filtered to attain the highest-quality dataset by adhering to the following commonly used criteria:

- **Step 1:** Protein sequences that have evidence "inferred from homology" for the existence of a protein were removed;
- **Step 2:** Protein sequences annotated with multiple functions (e.g., transporters and enzymes) were removed.
- **Step 3:** Protein sequences that have no `GO` MF annotation or annotations based solely on computational evidence (IEA) were eliminated;

- **Step 4:** Protein sequences with more than 60% pairwise sequence identity were removed via a CD-HIT [LG06] program to avoid any homology bias.

The dataset contained 2,224 transporter sequences. We then used the *Ontoclass* to find their substrate class. Of the 2,224 sequences, 1,524 had clear substrate annotations, 379 had no `ChEBI` mapping, and 321 had multiclass annotations. Table 25 indicates the number of proteins assigned to each class. The class with the largest number of proteins was 1.C (*inorganic cations*) with 601 transporters. The class with the second highest number was 5.A (*amino acids*) with 147 proteins, followed by 4.A (*monosaccharides and derivatives*) with 126 sequences. Classes 5.F (*proteins*), 2.B (*organic anions*), and 1.D (*inorganic anions*) had 113, 107, and 102 proteins, respectively.

5 Discussion

Building a substrate-specific transporter protein dataset requires assigning a substrate class to a transporter that represents the general substrate that the transporter transports across the membrane. Most of the established manually curated benchmark datasets extract this information from the `Swiss-Prot` database. The `Swiss-Prot` database is manually annotated and reviewed in terms of which `GO` terms (i.e., MF, BP, or CC) are assigned to the protein records, along with the reference from which the term was derived and the evidence code that indicates the degree to which the annotation is supported. The `GO` MF terms describe the activities that occur at the molecular level and are thus utilized in our tool; the `GO` terms are cross-referenced with other ontologies. Of specific interest to us is the `ChEBI` ontology, which provides information on chemical entities. There are 1,080 descendant terms of the `GO:0005215 transporter activity` annotation, 775 of which are cross-referenced with the `ChEBI` ontology. The cross-references of the descendant terms indicate the transported substrate. The advantage of using an ontology to collect the transported substrates is that the relationships between more specific substrates and broader classes are already established and therefore can be delegated to the ontology rather than to the dataset curator. Thus, any user who does not have significant prior knowledge of chemistry can build a new dataset or expand an already established one to accommodate the newly added entries.

We used the proposed *Ontoclass* tool to infer the substrate classes of two datasets. The first benchmark dataset was developed by Mishra et al. [MCZ14], wherein the substrate classes were manually assigned. *Ontoclass* was not able to infer the classes of approximately 28% of the sequences in this dataset due to insufficient GO annotations to assign the classes. Furthermore, we compared the manually assigned classes with those inferred by *Ontoclass* and discovered that the majority of the assigned classes (90%) are similar and that the disagreement arises from different interpretations between the dataset curator and the established ontology. Thus, using the standard performance measures to evaluate *Ontoclass* is not useful. In essence, we are trying to compare the manual curation conducted by Mishra et al. to the ChEBI ontology relationships, which are in turn manually annotated by expert annotators. For example, a transporter that transports *L-glutamate*, an α-amino acid anion, was mapped to the *organic anions* class in the ChEBI ontology, whereas the benchmark dataset assigned it to the *amino acids* class. In addition, a transporter that transports *phytochelatin* was mapped to the *peptides* class in the ChEBI ontology, whereas the benchmark dataset assigned it to the *other* class. An advantage of delegating the decision to the ontology is that, unlike the decisions made by a dataset curator, all of the reasoning is included, and thus, the decisions are easily reproduced.

The second dataset was extracted from the Swiss-Prot database and includes all of the proteins with transporter activity annotations. The goal of this case study is to achieve an overview of the available substrate classes. As expected, the distribution of substrates is not equal. The majority of mapped sequences belong to the 1.C *inorganic cations* substrate class, which is expected since ion channel transporters compose a large class that transports ions such as potassium, sodium, and calcium. Other classes include fewer sequences, even though the tool does not infer specific substrates (e.g., benzoates) and grouped them into larger categories (e.g., monocarboxylic acids); we acquired a small number of proteins from many classes. This phenomenon highlights the fact that membrane proteins are still not well characterized and that there is still insufficient data available to build a predictor that can predict the specific substrate.

6 Conclusion

The construction of benchmark datasets for supervised learning requires a label or class to be assigned to each datapoint. In those cases where the label is not directly taken from a reference source, this is done by the constructor of the dataset . In transporter substrate prediction, building a transporter dataset is commonly conducted through manual curation, in which the rationale behind assigning specific substrates to more general classes is not explained. The lack of documentation has created many challenges when establishing a new dataset or updating an established dataset. This chapter demonstrates that using transporter-related `GO` MF terms from the annotations in the `Swiss-Prot` database along with their corresponding `ChEBI` mappings can help to achieve automation. We have proposed an automated tool (*Ontoclass*) that exploits the well-defined and consistent annotation by the `Swiss-Prot` curators and delegates the substrate class to established ontologies without any external dataset curator judgment. The automated tool relieves us of the burden of manual curation to assign a label; it is consistent with other ontologies and is reproducible. It can adapt to the exponential growth and updates of biological databases with minimal prior knowledge.

Table 25: Number of sequences in each substrate class with similarity removed

Category and substrate type	**Class**	**Number of sequences**	**Total**
1. Inorganic molecules	A. Nonselective	26	755
	B. Water	26	
	C. Cations	601	
	D. Anions	102	
2. Organic ions	A.Organic cations	13	120
	B.Organic anions	107	
3. Carbohydrates and derivatives	A. Monosaccharides and derivatives	126	141
	B. Oligosaccharides and derivatives	11	
	C. Polysaccharides and derivatives	4	
4. Carboxylic acids	A. Monocarboxylic acids	28	33
	B. Tricarboxylic acids	4	
	C. Dicarboxylic acids	1	
5. Organonitrogen compounds	A. Amino acids	147	317
	B. Amino acid derivatives	10	
	C. Peptides	27	
	D. Amines	4	
	E. Polyamines	11	
	F. Proteins	113	
	G. Other organic amino compounds	5	
6. Organic heterocyclic compounds	A. Nucleobases	18	61
	B. Nucleosides	16	
	C. Nucleic acid s	3	
	D. Nucleotides	24	
7. Miscellaneous	A. Polyols	4	97
	B. Organic phosphates	23	
	C. Amides	3	
	D. Other organic molecular entities	67	

This table shows the distribution of substrate classes in the `Swiss-Prot` database for all of the sequences that have `GO` MF `GO:0005215 transporter activity` annotations with less than 60% similarity.

Chapter 7

Predicting substrate specificity

This chapter addresses the fourth research objective:

O4: To broaden the scope of the state-of-the-art for substrate class prediction while maintaining credible predictive performance.

Some parts of this chapter have been published in *PLoS ONE*: Alballa, M., Aplop, F., & Butler, G. (2020). TranCEP: Predicting the substrate class of transmembrane transport proteins using compositional, evolutionary, and positional information. *PLoS ONE*, 15(1), e0227683.

Contributions of the authors are as follows:

- **Alballa, M.** Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Review & editing.
- **Aplop, F.** Conceptualization, Review & editing.
- **Butler, G.** Conceptualization, Formal analysis, Methodology, Project administration, Supervision, Writing – original draft, Review & editing.

The chapter is organized as follows: Section 1 provides an introduction to transporter substrate prediction. Section 2 describes the materials and methods utilized to build the proposed tool, *TooT-SC*. Section 3 presents and analyzes the results, and Section 3.2 compares the performance of *TooT-SC* with the state-of-the-art methods. Finally, Section 4 concludes the chapter.

1 Introduction

Existing tools for the annotation of transporters that predict the substrates of transport reactions lag behind tools for other kinds of proteins, such as for predicting enzymes involved in metabolic reactions.

Many tools rely simply on homology or orthology to predict transporters. These tools include the metabolic network tools merlin [DRFR15], Pantograph [LZS15], and TransATH [AB17].

Among the tools for *de novo* prediction of substrate class, FastTrans [HPO+19] claims to be the state-of-the-art. *De novo* prediction tools predict the type of substrate from a general subset of substrate types, without attempting to predict the specific substrate [SCH10, COLG11, SH12, BH13, MCZ14]. The main reason for this is that the current number of annotated transporters with specific substrates is still quite limited. As discussed in Chapter 6, even when the transporter substrates of the `Swiss-Prot` database are grouped into a higher level of abstraction, there are still a small number of samples in many classes. This phenomenon hinders the possibility of building tools that predict exact substrates. Tools that classify transporters based on their substrate specificity have reached a maximum of seven substrate types [MCZ14] [HPO+19]. For network modeling in systems biology [TP10,SAJT14], we require tools to process the complete proteome and predict each transport reaction, which means identifying the transport protein and the specific substrate.

Our laboratory's previous efforts for the *de novo* prediction of specific substrates for sugar transporters in fungi were not successful [Apl16]. However, from these studies, we learned how much depends on a very few residues of the transporter, often approximately three residues, and often internal to different helix TMSs of the transporter [FBS+14]. These residues are far apart in the linear protein sequence but close to each other in the 3D structure of the protein when integrated in the membrane. In looking forward to how we may improve upon approaches that rely on the amino acid composition of the protein, we developed a protocol whereby the compositional information is combined with evolutionary information as captured by an MSA, and by positional information on the residues responsible for determining the specificity of the transporter [TWNB12]. This protocol is a schema for a large number of possible algorithms, due to the many choices for encoding amino acid compositions, MSA algorithms, and algorithms for specificity-determining sites [CC14]. We also realized the importance of the alignment in preserving the TMS positions because the important residue positions seem to be located there. There are a number of such MSA algorithms [PFH08, CDTTN12, FTC^{+}16, BGA^{+}17].

We therefore conducted a study utilizing a new benchmark dataset *DS-SC* with additional substrate classes, which indicated that the combination of information about protein composition, protein evolution, and the specificity-determining positions had a significant impact on our ability to predict the transported substrates. We chose the TrSSP methodology [MCZ14] as our baseline and varied it to illustrate the impact of each of the factors: compositional, evolutionary, and positional information. Our best approach, which defines our predictor ,*TooT-SC*, involves utilizing the PAAC encoding scheme, the TM-Coffee MSA algorithm [CDTTN12], and the transitive consistency score (TCS) algorithm [CDTN14] for determining informative positions in the MSA to build a suite of SVM classifiers, one for distinguishing each substrate class.

2 Materials and methods

2.1 Dataset

To construct a high-quality benchmark dataset, the `UniProt`-IDs from the transporter dataset in Chapter 5 were used as input to our ontology-based tool *Ontoclass* to assign a substrate class to the transporters. This was then modified in such a way that the sequences that belong to the substrate classes with very small samples were grouped into a higher level of abstraction. The final dataset contains 11 substrate classes, with the largest being the *inorganic cations* class with 601 samples and the smallest being the *nucleotide* class with 24 samples, as presented in Table 26. The data were randomly partitioned (stratified by class) into training (90%) and testing (10%) sets. We refer to the data in Table 26 as *DS-SC.* To the best of our knowledge, these data contain the highest number of substrate classes being used to predict the substrate class of a transporter.

Table 26: Dataset *DS-SC*

ID	Substrate class	Training	Testing	Total
C1	Nonselective	24	2	26
C2	Water	24	2	26
C3	Inorganic cations	541	60	601
C4	Inorganic anions	92	10	102
C5	Organic anions	97	10	107
C6	Organo-oxygens	157	17	174
C7	Amino acids and derivatives	142	15	157
C8	Other organonitrogens	144	16	160
C9	Nucleotides	22	2	24
C10	Organic heterocyclics	34	3	37
C11	Miscellaneous	99	11	110
Total		**1,376**	**148**	**1,524**

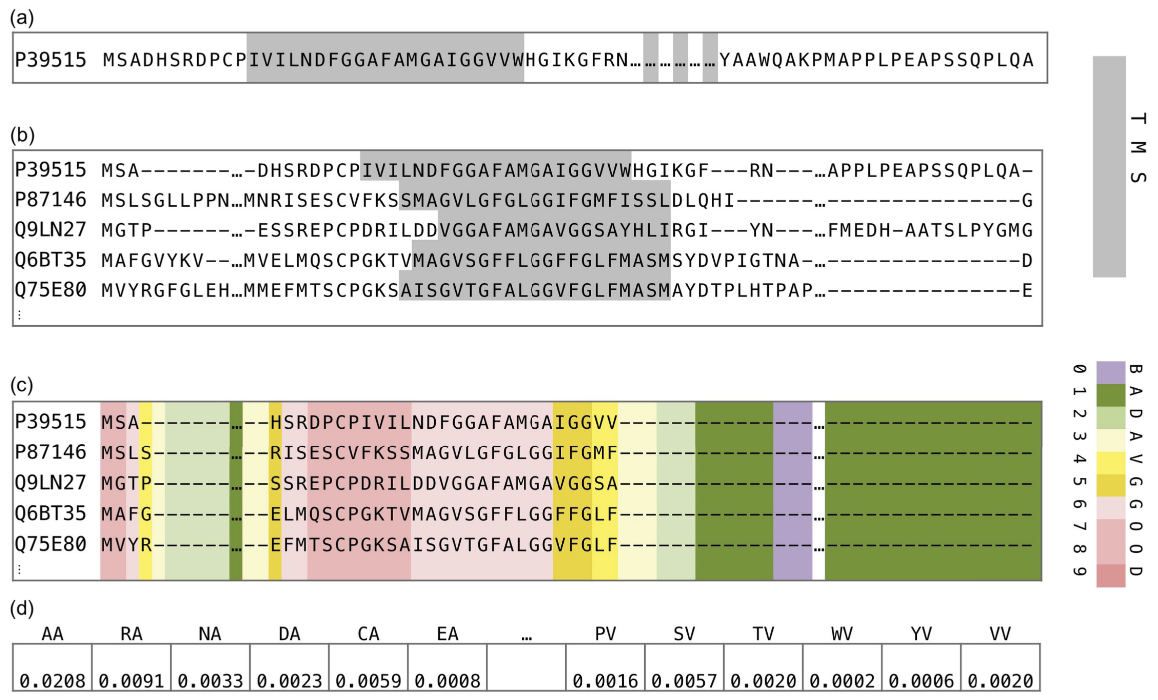
2.2 Databases

We used the `Swiss-Prot` database when searching for similar sequences. When constructing MSAs, we used TM-Coffee [CDTTN12] with the `UniRef50-TM` database, which consists of the entries in `UniRef50` that have the keyword *transmembrane*.

2.3 Algorithm

Figure 21 illustrates the steps of the *TooT-SC* method. The sequence in (a) has four TMSs, as shown by the gray shading. The example focuses on the first TMS and abbreviates the middle section of the sequence. Part (b) shows an MSA conserving the TMS structure constructed by TM-Coffee, where the gray shading indicates the TMS location. Part (c) shows the color coding of the reliability index of each column as determined by TCS, and shows how gaps replace unreliable columns in the filtered MSA. Part (d) shows a part of the 400-dimensional vector of dipeptide frequencies (PAAC) from the filtered MSA.

The template for combining evolutionary, positional, and compositional information is presented in Algorithm 4. Note that the use of evolutionary (E) and positional (P) information is optional, and that if positional (P) information is used, then it requires evolutionary (E) information in the form of MSA. Note also that if Step (E) is not completed, then the compositional Step (C) encodes the sequence s. Finally, note that if Step (E) is completed but Step (P) MSA is not, then Step (C) encodes the MSA.



```
P39515  MSADHSRDPCPIVILNDFGGAFAMGAIGGVVWHGIKGFRN……………YAAWQAKPMAPPLPEAPSSQPLQA
```

```
P39515  MSA-------…-DHSRDPCPIVILNDFGGAFAMGAIGGVVWHGIKGF---RN---…APPLPEAPSSQPLQA-
P87146  MSLSGLLPPN…MNRISESCVFKSSMAGVLGFGLGGIFGMFISSLDLQHI------…---------------G
Q9LN27  MGTP------…-ESSREPCPDRILDDVGGAFAMGAVGGSAYHLIRGI---YN---…FMEDH-AATSLPYGMG
Q6BT35  MAFGVYKV--…MVELMQSCPGKTVMAGVSGFFLGGFFGLFMASMSYDVPIGTNA-…---------------D
Q75E80  MVYRGFGLEH…MMEFMTSCPGKSAISGVTGFALGGVFGLFMASMAYDTPLHTPAP…---------------E
⋮
```

```
P39515  MSA-------…--HSRDPCPIVILNDFGGAFAMGAIGGVV----------------…----------------
P87146  MSLS------…--RISESCVFKSSMAGVLGFGLGGIFGMF----------------…----------------
Q9LN27  MGTP------…--SSREPCPDRILDDVGGAFAMGAVGGSA----------------…----------------
Q6BT35  MAFG------…--ELMQSCPGKTVMAGVSGFFLGGFFGLF----------------…----------------
Q75E80  MVYR------…--EFMTSCPGKSAISGVTGFALGGVFGLF----------------…----------------
⋮
```

AA	RA	NA	DA	CA	EA	…	PV	SV	TV	WV	YV	VV
0.0208	0.0091	0.0033	0.0023	0.0059	0.0008		0.0016	0.0057	0.0020	0.0002	0.0006	0.0020

Figure 21: Example of steps of the *TooT-SC* method

The figure illustrates the steps of the *TooT-SC* method. Note that we abbreviated the middle section of the sequence. Part (a) shows the sequence of the four TMSs in gray. Part (b) shows an MSA constructed by TM-Coffee. The gray shading indicates a TMS. Part (c) shows the color coding of the reliability index of each column as determined by the TCS, as well as gaps in unreliable columns in the filtered MSA. Part (d) shows a 400-dimensional PAAC vector from the filtered MSA.

Algorithm 4 Template for constructing the composition vector

```
function COMP_VEC(seq $s$)
    // Evolutionary (E) step, optional
    Construct an MSA from $s$
    // Positional (P) step, optional
    Determine the informative positions (columns) in the MSA
    Filter the uninformative positions from the MSA
    // Compositional (C) step, mandatory
    return Vector-encoding composition of the filtered MSA
end function
```

In this work, we used TM-Coffee to compute the MSA that conserves the TMSs and the TCS to determine a reliability index for each position (column) in the MSA. We experimented with three composition schemes, AAC, PAAC, and PseAAC, as well as the optional use of TM-Coffee and the TCS. Algorithm 5 shows the composition vectors being used to build a set of classifiers (SVM classifiers in this case). Algorithm 6 presents the prediction algorithm.

Algorithm 5 Building the SVM classifiers

```
Require: training set $T$ of sequences labeled with classes $C_1, ..., C_n$
Ensure: set of SVMs $svm(i)$, distinguishing class $C_i$ from other classes
  procedure BUILD_SVMS($T$: a set of seqs; $svm$: a set of SVMs)
    for all seq $s$ in $T$ do
      $v(s) \leftarrow$ COMP_VEC( $s$ )
    end for
    for all $(C_i)$ in classes do
      $C_{\hat{i}} : \{C_1, ..., C_n\} - C_i$
      $svm(i) \leftarrow$ SVM.build($\{v(s) : s \in T \cap (C_i \cup C_{\hat{i}})\}$, probability= T)

    end for
  end procedure
```

Algorithm 6 Prediction

```
Require: test sequence $s$
Require: set of SVMs $svm(i)$ distinguishing classes $C_i$ from other classes
Ensure: result is the predicted class $C_p$
  function PREDICT_CLASS(seq $s$)
    $v \leftarrow$ COMP_VEC( $s$ )
    $c \leftarrow$ array of length $n$
    for all $C_i$ in $\{C_1, ..., C_n\}$ do
      $c[i] \leftarrow$ probability of class $i$ ($svm(i)$ applied to $v$)
    end for
    $C_p \leftarrow argmax(c)$
    return $C_p$
  end function
```

2.4 Encoding the amino acid composition

The properties of the amino acids at each position in the protein sequence can be encoded into vectors that summarize the overall composition of the protein. Three approaches for encoding the amino acid composition were implemented in this study: AAC, PAAC, and

PseAAC.

2.5 MSA

We adopted the MSA-AAC approach [SCH10] that combines AAC with the evolutionary information available from the MSA.

This method is implemented by first retrieving homologous sequences of each protein sequence in the dataset, building an MSA for the corresponding protein, and then taking the counts for computing the composition information using all of the residues in the MSA. Schaadt et al. [SCH10], utilized only AAC encoding, whereas we also applied the approach to PAAC and PseAAC encoding. Another difference was that we made use of TM-Coffee [CDTTN12] (Version-11.00.8cbe486) to compute the alignments, rather than ClustalW [THG94], as was done by Schaadt et al. [SCH10], because we felt it was important to align the TMSs.

Other differences included searching the `Swiss-Prot` database [BBA+03] and retrieving a maximum of 120 homologous sequences instead of searching the nonredundant database `nr` and retrieving 1,000 sequences. This process was done to make the computational time more manageable because the TM-Coffee algorithm requires a great deal of memory and a longer execution time.

Furthermore, all exact hits of the test sequences were removed from the `Swiss-Prot` and `UniRef50-TM` databases, to maintain a degree of independence between the MSA and the test data. It should be noted that any bias in `Swiss-Prot` is still inherited by both the training data and the test set; however, the last step reduces correlation. Our alignment command was the following:

```
t_coffee mysequences.fasta -mode psicoffee \
-protein_db uniref50-TM \
-template_file PSITM
```

where `mysequences.fasta` is the file that contains the 120 similar sequences retrieved by a BLAST search on the `Swiss-Prot` database.

2.6 Positional information

To focus on those positions in the protein that determine the specificity, we needed a method to determine those positions, and then to filter our MSA. The MSA was filtered by setting the entries for all other positions to null, that is, the symbol "–" so that it was ignored when gathering the counts for the amino acid composition.

We applied the TCS algorithm [CDTN14] to the alignment to determine the informative positions. The TCS is a scoring scheme that 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 were removed using the following command:

```
t_coffee -infile myMSA.aln -evaluate \
-output tcs_column_filter4.fasta
```

where `myMSA.aln` is the MSA file and `tcs_column_filter4.fasta` is the filtered file in FASTA format.

2.7 Training

Following TrSSP [MCZ14], we used SVM with an RBF kernel, as implemented in the `R` *e1071* library (version 1.6-8), utilizing a one-against-the-rest approach in which n binary classifiers are trained, one for each class. The classifier i is trained with all the samples of class i as a positive class and the rest as a negative class. The final predicted class is the class with the highest probability among the n predictions. Both the cost and γ parameters of the RBF kernel were optimized by performing a grid search using the *tune* function in the library (cost range: $2^{(1...5)}$, γ range: $2^{(-18...2)}$).

2.8 Methods

We adopted three approaches to encode the amino acid composition: AAC, PAAC (as done by TrSSP [MCZ14]), and PseAAC. This was followed by training using an SVM to form the prediction methods **AAC**, **PAAC**, and **PseAAC**.

By combining the amino acid composition and the evolutionary information obtained using TM-Coffee, followed by an SVM, we implemented the prediction methods: **TMC-AAC**, **TMC-PAAC**, and **TMC-PseAAC**.

Filtering was incorporated by applying TCS after TM-Coffee, computing the amino acid composition vectors, and applying the SVM to implement the prediction methods: **TMC-TCS-AAC**, **TMC-TCS-PAAC**, and **TMC-TCS-PseAAC**.

The method used in ***TooT-SC*** is **TMC-TCS-PAAC**—the method that achieved the best performance during cross-validation.

2.9 Performance evaluation

The performance of each method on the *DS-SC* training set was determined using 10-CV, whereby the training dataset was randomly partitioned into ten sets of equal size. A single set was kept as the validation data, and the remaining nine sets were used to train the SVM model. This model was then tested using the validation set. The cross-validation process was repeated nine times, where each of the sets was used once as the validation data. The performance of all the models was aggregated and used to produce a single estimate (microaverage).

Since the 10-CV performance varies with different random splits and to make the error estimation more stable, we repeated the 10-CV process ten times with different random partitions, and the performance variations between the runs were captured by computing the standard deviation. It has been reported [Koh95] that this repetition stabilizes the error estimation and therefore reduces the variance in the K-CV estimator. Throughout the rest of this chapter, the cross-validation performance is reported as the $mean$s ± SDs of the ten different runs of the 10-CV process.

Four statistical measures were considered to measure the performance:

The sensitivity, which is the proportion of positive samples that are correctly identified:

$$Sensitivity = \frac{TP}{TP + FN} \tag{44}$$

The specificity, which is the proportion of negative samples that are correctly identified:

$$Specificity = \frac{TN}{TN + FP} \tag{45}$$

The accuracy, which is the proportion of correct predictions made among all the predictions:

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{46}$$

The MCC, which is a single measure taking into account the true and false positives and negatives:

$$MCC = \frac{(TP \times TN - FP \times FN)}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \tag{47}$$

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

We used the MCC because it is less influenced by imbalanced data and is arguably the best single assessment metric in this case [Din11, WP03, BDA13]. The MCC value ranges from 1 to -1, where 1 indicates a perfect prediction, 0 represents no better than random, and -1 implies total disagreement between the prediction and observation. A high MCC value means that the predictor has high accuracy on both positive and negative classes and low misclassification in both classes.

When dealing with multiclass classification, it is often desirable to compute a single aggregate measure that reflects the overall performance. There are two methods to compute the overall performance, namely, *microaveraging* and *macroaveraging* [MRS08]. Macroaveraging computes the simple average performance of individual class performances. Microaveraging computes the overall performance by globally counting the total true positives, false negatives and false positives. Depending on the class distribution, the difference between the two methods can be large. Macroaveraging gives equal weight to each class, whereas microaveraging gives equal weight to each individual classification decision [MRS08]. The overall accuracy of the tool is often calculated as the fraction of the correct predictions by the total number of predictions as follows:

$$Accuracy_{overall} = \sum_{k=1}^{K} \frac{TP_k}{N} \tag{48}$$

where TP_k is the number of true positives in class k, K is the number of different classes, and N is the total number of predictions.

Another way to compute the accuracy is to take the macroaverage accuracy of the individual classes:

$$Accuracy_{macro} = \frac{1}{K} \sum_{k=1}^{K} Accuracy_k \tag{49}$$

where $Accuracy_k$ is the accuracy of class k, and K is the number of different classes.

Similarly, the overall MCC is calculated in terms of a $K \times K$ confusion matrix C [Gor04]:

$$MCC_{overall} = \frac{\sum_k \sum_l \sum_m C_{kk}C_{lm} - C_{kl}C_{mk}}{\sqrt{\sum_k (\sum_l C_{kl})(\sum_{k'|k' \neq k} \sum_{l'} C_{k'l'})}\sqrt{\sum_k (\sum_l C_{lk})(\sum_{k'|k' \neq k} \sum_{l'} C_{l'k'})}} \tag{50}$$

Or as a macroaverage MCC:

$$MCC_{macro} = \frac{1}{K} \sum_{k=1}^{K} MCC_k \tag{51}$$

where MCC_k is the accuracy of class k, and K is the number of different classes. Because the number of samples in each class of the dataset is imbalanced, we used the overall accuracy as in Equation 48 and the overall MCC as in Equation 50 to evaluate and compare the different methods. It is explicitly stated when the macroaverage was used.

2.10 Statistical analysis

In this analysis, Student's (two-tailed, paired) t-tests were applied, and the average number of informative residues, as determined by TCSs, in different segments of a protein sequence was computed. For each substrate class, pairwise comparisons between the means of important positions in different segments were performed. The differences were considered statistically significant when the P-value of the Student's t-test was less than 0.0001.

3 Results and discussion

3.1 Methods evaluation

Since the data are imbalanced, we focused on the MCC when comparing the performances of the different models. Table 27 presents the overall accuracy values and MCCs of the SVM models for the nine methods, sorted from the best to the worst according to the MCC. The details of the performance for each method are available in Appendix C; the comparisons among the different methods for the eleven classes in terms of the MCC are presented in Figure 22.

Table 27: Overall cross-validation performance of the methods

Method	**Accuracy**	**MCC**
TMC-TCS-PAAC	82.53 ± 0.12	0.7772 ± 0.0019
TMC-PAAC	81.92 ± 0.12	0.7695 ± 0.0014
TMC-AAC	79.84 ± 0.13	0.7430 ± 0.0014
TMC-PseAAC	79.46 ± 0.30	0.7374 ± 0.0038
TMC-TCS-AAC	79.33 ± 0.24	0.7360 ± 0.0035
TMC-TCS-PseAAC	79.03 ± 0.27	0.7324 ± 0.0037
PAAC	58.93 ± 0.45	0.4610 ± 0.0069
PseAAC	54.80 ± 0.76	0.3999 ± 0.0108
AAC	52.21 ± 0.60	0.3628 ± 0.0091

For each method, the table presents the accuracy and MCC as the *means* ± *SD*s across the ten runs of the 10-fold cross-validation.

The SVM model that utilized PAAC encoding outperformed those that utilized AAC and PseAAC encoding by 27% and 15%, respectively, in terms of the overall MCCs. This model shows exceptionally high performance in the *water* and *nucleotide* classes. In addition, all of the SVM models that utilized evolutionary data performed notably better overall than the SVM models that did not. The top model, **TMC-TCS-PAAC**, which is the method chosen for our predictor ***TooT-SC***, incorporates the use of the PAAC with evolutionary data in the form of MSA with positional information, in which columns that have a reliability below 4 are filtered out. We found that the performance peaked using this threshold and

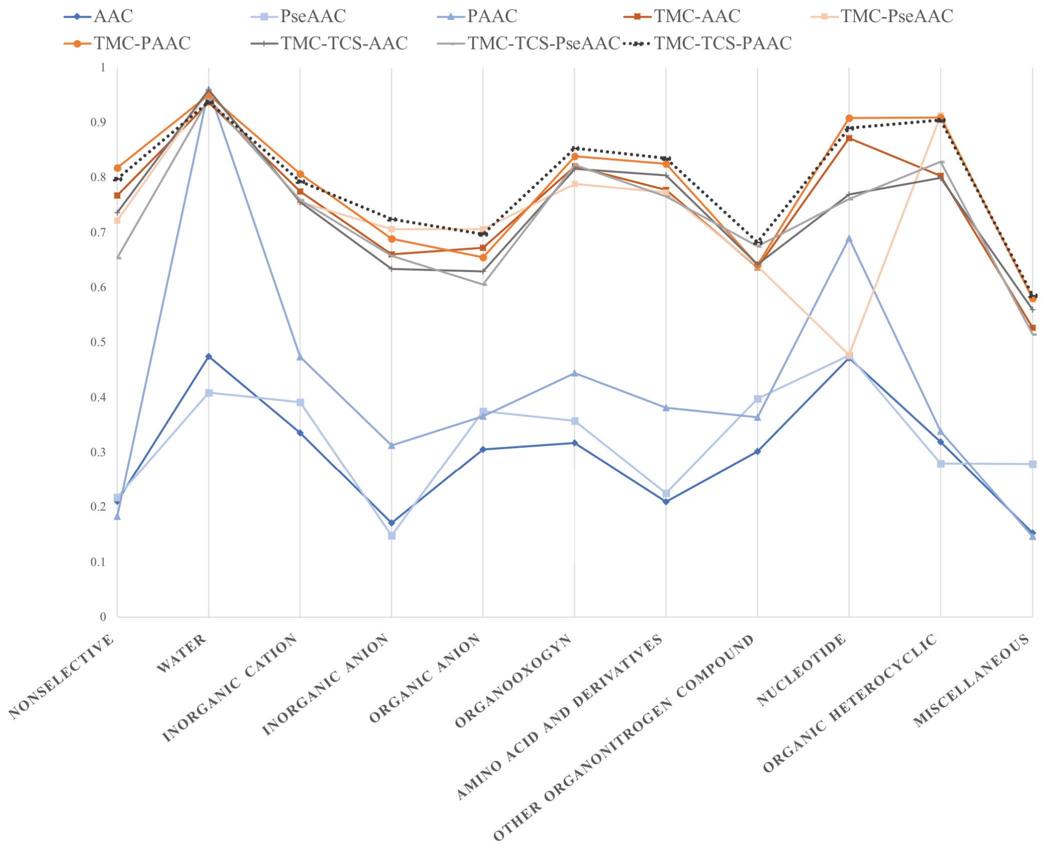


Figure 22: MCCs of the different methods on the substrate classes
This figure shows the cross-validation MCC performance of the different methods on the eleven substrate classes. The dotted line represents the performance of *TooT-SC*, which is TCS-TMC-PAAC

started to decline when columns with a reliability index greater than 4 were filtered out. The **TMC-TCS-PAAC** method yielded an overall MCC of 0.77 during cross-validation. Table 29 shows the impact of evolutionary information and positional information on the composition-encoding PAAC.

The use of evolutionary information in the form of MSA on the composition-encoding PAAC showed a considerable positive impact in most of the substrate classes, where the average improvement of the MCC was 126.41%, with the highest improvement being in the C1 (*nonselective*) class (347%). The baseline encoding PAAC for the C2 (*water*) substrate class showed a high discriminatory power with an MCC of 0.96, with the incorporation of additional information having a slightly negative impact of 1.01%.

The further use of positional information by filtering out the unreliable columns from the

Table 28: TMC-TCS-PAAC performance

Class ID	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
C1	75.00 ± 0.00	99.78 ± 0.00	99.21 ± 0.00	0.7979 ± 0.0000
C2	95.83 ± 0.00	99.85 ± 0.00	99.74 ± 0.00	0.9376 ± 0.0000
C3	95.19 ± 0.47	86.92 ± 0.28	89.36 ± 0.21	0.7936 ± 0.0046
C4	64.35 ± 1.97	99.24 ± 0.18	96.38 ± 0.19	0.7252 ± 0.0155
C5	68.04 ± 0.49	98.40 ± 0.13	95.66 ± 0.14	0.6974 ± 0.0084
C6	83.44 ± 0.52	98.97 ± 0.12	96.72 ± 0.15	0.8543 ± 0.0066
C7	84.08 ± 0.95	98.55 ± 0.16	96.56 ± 0.18	0.8357 ± 0.0085
C8	71.46 ± 0.95	96.84 ± 0.27	93.42 ± 0.22	0.6830 ± 0.0084
C9	80.91 ± 1.92	99.98 ± 0.04	99.61 ± 0.05	0.8904 ± 0.0132
C10	82.35 ± 0.00	100.00 ± 0.00	99.47 ± 0.00	0.9050 ± 0.0000
C11	55.96 ± 1.09	97.95 ± 0.16	94.21 ± 0.20	0.5858 ± 0.0136
Overall			82.53 ± 0.12	0.7772 ± 0.0019

Table 29: Impact of factors on the performance of the PAAC.

Class ID	**MCC**			**TMC-PAAC to PAAC**		**TMC-TCS-PAAC to PAAC**		**TMC-TCS-PAAC to TMC-PAAC**	
	PAAC	TMC-PAAC	TMC-TCS PAAC	Delta	%	Delta	%	Delta	%
C1	0.18	0.82	0.80	0.64	347.27	0.61	336.01	-0.02	-2.52
C2	0.96	0.95	0.94	-0.01	-1.01	-0.02	-2.40	-0.01	-1.41
C3	0.47	0.81	0.79	0.33	70.12	0.32	67.25	-0.01	-1.68
C4	0.31	0.69	0.73	0.38	120.32	0.41	131.69	0.04	5.16
C5	0.37	0.66	0.70	0.29	78.83	0.33	90.23	0.04	6.38
C6	0.44	0.84	0.85	0.40	88.82	0.41	92.11	0.01	1.74
C7	0.38	0.83	0.84	0.44	116.44	0.45	119.06	0.01	1.21
C8	0.36	0.64	0.68	0.28	75.77	0.32	87.23	0.04	6.52
C9	0.69	0.91	0.89	0.22	31.72	0.20	29.02	-0.02	-2.05
C10	0.34	0.91	0.91	0.57	168.32	0.57	166.96	0.00	-0.51
C11	0.15	0.58	0.59	0.43	293.97	0.44	297.15	0.00	0.81
Average				0.36	126.41%	0.37	128.57%	0.01	1.24%

This table notes the differences in the MCC, delta, percentage improvement in the MCC, and the percent of the cross-validation performance for the introduction of evolutionary information using TM-Coffee, and the positional information using the TCS. The use of evolutionary information in the form of an MSA on the composition-encoding **PAAC** improved the MCC by an average of 126.41%. The further use of positional information by filtering out the unreliable columns from the MSA boosted the MCC of the composition encodings by an average of 128.57%.

MSA showed an average improvement of 128.57% compared to the baseline compositions. The impact of positional information over that already achieved by evolutionary information showed a positive impact in most substrate classes; the highest was in the C5 (*organic anions*) class, where the MCC improved by 6.38% with **TMC-TCS-PAAC**. However, the impact was slightly negative in the C1 (*nonselective*), C2 (*water*), C3 (*inorganic cations*), and C9 (*nucleotides*) classes.

3.2 Comparison with other published work

The top two tools with the best reported performance are TrSSP [MCZ14] and FastTrans [HPO$^+$19]. Since the original code was not available for TrSSP or FastTrans, we reimplemented the methods to the best of our ability. We compared the performance of the *TooT-SC* method with our implementation of the TrSSP and FastTrans methods. All of the methods were trained using the *DS-SC* training set (Section 2.1) and tested using its testing set. It should be noted that our implementation of the TrSSP method [MCZ14] achieved a similar macroaverage MCC to that reported in the original paper (0.41) on their dataset. However, it was not possible to reproduce the reported performance of the FastTrans method [HPO$^+$19], for which our implementation on their same dataset achieved a macroaverage MCC of 0.47, while their reported macroaverage MCC was 0.87.

A comparison between the *TooT-SC* method and our implementation of the other state-of-the-art methods on the *DS-SC* benchmark dataset is presented in Table 30. The *TooT-SC* method scored higher than the other methods for all of the substrate classes in terms of the accuracy, sensitivity, and MCC. Overall, the *TooT-SC* method scored an overall MCC of 0.82, which outperformed the TrSSP method by 26% and the FastTrans method by 115%.

3.3 Positional information analysis

It is difficult to isolate the exact residues that are key to inferring the substrate class; the results suggest that evolutionary information, obtained by MSA, is the main source for achieving a high prediction performance. In addition, the TCS informative positions (with TCSs ≥ 4) can help to filter out unnecessary noise and obtain a clearer signal to further improve the prediction. Using the TCS informative positions filtered out an average of 31% $\pm$ 19% of the sequence. However, when we attempted to filter out more positions (by using a TCS score cutoff stricter than 4), the performance started to deteriorate.

To visualize the informative positions relative to the hydropathy scale of amino acids, the hydropathy scale proposed by [KD82] was utilized, and the average hydropathy of each

Table 30: Comparison between *TooT-SC* and the state-of-art methods

Class	**Specificity**			**Sensitivity**			**Accuracy**			**MCC**		
ID	TrSSP	FastTrans	TooT-SC	TrSSP	FastTrans	TooT-SC	TrSSP	FastTrans	TooT-SC	TrSSP	FastTrans	TooT-SC
C1	100.00	100.00	100.00	0.00	0.00	50.00	98.18	97.70	99.22	0.00	0.00	0.70
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	76.14	88.64	91.67	86.67	96.67	83.08	74.56	91.37	0.68	0.50	0.83
C4	98.55	95.65	100.00	60.00	40.00	70.00	94.74	87.63	97.69	0.64	0.33	0.83
C5	98.55	97.83	97.83	80.00	50.00	90.00	96.43	91.40	96.95	0.78	0.51	0.81
C6	96.95	96.18	97.71	64.71	35.29	76.47	91.53	84.16	94.78	0.64	0.35	0.76
C7	97.74	87.97	100.00	73.33	40.00	86.67	93.91	77.27	98.45	0.72	0.20	0.92
C8	94.70	96.21	96.21	56.25	25.00	87.50	88.52	83.33	94.78	0.50	0.25	0.77
C9	99.32	99.32	100.00	100.00	0.00	100.00	99.08	96.59	100.00	0.81	-0.02	1.00
C10	98.62	100.00	100.00	33.33	66.67	100.00	96.43	98.84	100.00	0.31	0.81	1.00
C11	99.27	95.62	100.00	27.27	36.36	45.45	92.31	86.73	95.49	0.42	0.31	0.66
Overall							72.97	57.43	85.81	0.65	0.44	0.82
Macroaverage							93.94	88.93	97.16	0.57	0.39	0.84

This table presents the performance of the proposed tool (*TooT-SC*) built with the complete training set and run on the independent testing set of *DS-SC* (see Table 26) and the corresponding results for the TrSSP and FastTrans methods trained and tested with the same dataset. This table shows the 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 as in Equation 50.

column in the MSA was computed. Higher positive scores indicate that amino acids in that region have hydrophobic properties and are likely located in a transmembrane α-helix segment. The TCS of each column in the alignment is noted on the hydropathy plot through color coding. Figure 23 shows diverse examples. The red shades correspond to the informative columns (TCS $\geq$ 4), while the gray and white shades correspond to noninformative columns that are filtered out by *TooT-SC*. In Figure 23 (a) and (b), the regions with high positive average hydropathy values appear to be more informative than those with lower values. However, in Figure 23 (c) and (d), the difference between the informative positions with high and low hydropathy values is not as clear.

To measure the informative positions relative to different segments of the protein sequence, we divided the protein sequence positions into those in the TMS and those not in the TMS. Those in the TMS were divided into the interior one-third positions, and the remaining exterior positions in the TMS. The non-TMS positions were divided into those near a TMS, that is, within 10 positions, and the remaining positions were considered far from a TMS. The location of the TMS was retrieved from the `Swiss-Prot` database under the subcellular location topology section. Table 31 shows a breakdown of where the informative positions, as determined by the TCS, are located with respect to the TMS regions.

Table 31: Positional information.

Class ID	**SeqLth**	**TMS**	**TMSLth**	**Positions**		**TMS**			**Non-TMS**		
				Num	%Seq	Num	Interior Num	Exterior Num	Num	Close Num	Far Num
C1	322	4	81	200	64.35	63	22	41	138	35	103
C2	273	6	126	203	74.72	121	42	79	82	57	25
C3	681	7	149	387	57.23	126	45	81	250	65	185
C4	575	8	168	376	62.01	142	50	92	215	73	142
C5	598	10	203	417	70.69	179	62	117	233	91	142
C6	461	10	203	325	70.45	177	62	115	144	70	74
C7	467	10	206	306	67.33	170	59	111	136	83	53
C8	537	4	83	347	39.34	70	24	46	133	37	96
C9	403	6	129	282	71.25	122	43	79	159	79	80
C10	497	12	241	402	79.86	218	76	142	183	95	88
C11	639	7	149	349	47.44	110	38	72	164	54	110

This table presents information on the sites retained by the TCS filtering step. For each class of substrates in the dataset, the table presents the average sequence length (**SeqLth**), the average number of TMS regions (**TMS**), and the average total number of residues in the TMS regions (**TMSLth**). It also presents the average of the number of positions retained by the filtering step (**Positions: Num**) and the average of the number as a percentage of the total sequence length (**Positions: %Seq**). It notes the total number of sites that occur in the TMS regions (**TMS: Num**) and the non-TMS regions (**non-TMS: Num**). For the TMS regions, it presents the average number of informative sites that occur in the central one-third of the TMS regions (**TMS: Interior: Num**), and in the remaining exterior regions outside of the central one-third of the TMS regions (**TMS: Exterior: Num**). For the non-TMS regions, it presents the average number of informative sites that occur close to the TMS regions (within 10 positions of the TMS) (**non-TMS: Close: Num**) and the remaining sites far from the TMS regions (**non-TMS: Far: Num**).

For instance, in Figure 23 (a), 41.04% of the residues of the sequence with `UniProt`-ID `Q59NP1` are informative (i.e., correspond to informative columns in the alignment); thus, 58.96% of this sequence is filtered out. In this case, the residues in the TMSs of this protein are indeed more informative than those of the other proteins, where 100% of them are informative. On the other hand, only 29.19% of the residues in non-TMSs are informative.

The difference is not as significant in the sequence with `UniProt`-ID `Q9NY37` in Figure 23 (c), where the informative positions in the TMSs are similar to those of non-TMS positions. Details of the sequences in the figure are presented in Table 32.

Table 32: Examples of the informative residue distributions with respect to TMSs and non-TMSs

`UniProt`-ID	**SeqLth**	**TMS**	**TMSLth**	**Positions**		**TMS**		**non-TMS**	
				Num	% Seq	Num	% Seq	Num	% Seq
`Q59NP1`	251	2	42	103	41.04	42	100.00	61	29.19
`Q8BFW9`	622	12	252	386	62.06	246	97.62	140	37.84
`Q9NY37`	505	2	42	355	70.30	31	73.81	324	69.98
`Q9Y584`	194	3	63	78	40.21	32	50.79	46	35.11

This table shows the details of individual sequences in Figure 23. The table presents the sequence length (**SeqLth**), the number of TMS regions (**TMS**), and the total number of residues in the TMS regions (**TMSLth**). It also presents the number of informative positions retained by the filtering step (**Positions: Num**) and that number as a percentage of the total sequence length (**Positions: % Seq**). It also denotes the total number of informative sites that occur in the TMS regions (**TMS: Num**), as well as that number as a percentage of the total TMS length (**TMS: % Seq**). In addition, the total number of informative sites that occur in the non-TMS regions (**non-TMS: Num**) are reported, as well as that number as a percentage of the total non-TMS length (**non-TMS: % Seq**).

Table 33 presents a pairwise comparison between informative positions in the TMS and non-TMS regions. The sequences in all of the substrate classes except the C1 (*nonselective*) substrate class have significantly more informative positions in the TMS regions than in the non-TMS regions. Similarly, there is a significant difference between the informative positions close to TMSs and positions far from TMSs in all sequences that belong to all substrate classes except the C1 (*nonselective*) and C8 (*other organonitrogens*) classes, as shown in Table 34. In contrast, there is no difference between the informative positions in the central one-third of the TMS regions and the remaining exterior regions in the sequences that belong to the C1 (*nonselective*), C2 (*water*), C5 (*organic anions*), C8 (*other organonitrogens*), C9 (*nucleotides*), C10 (*organic heterocyclics*), and C11 (*miscellaneous*) classes; the difference is significant in the sequences that belong to the C3 (*inorganic cations*), C4 (*inorganic anions*), C6 (*organo-oxygens*), and C7 (*amino acids and derivatives*) classes, as presented in Table 35.

Table 33: Statistical analysis of the informative position rates in the TMS and non-TMS regions

Class ID	**TMS**	**non-TMS**	**P-value**
C1	80.74±23.46	58.69±22.43	0.0007
C2	95.58±9.43	57.48±12.14	<0.0001
C3	78.31±28.07	49.57±22.49	<0.0001
C4	79.81±27.38	53.94±25.36	<0.0001
C5	88.74±20.17	60.55±19.79	<0.0001
C6	85.35±15.54	56.20±16.58	<0.0001
C7	81.95±16.90	55.28±17.58	<0.0001
C8	46.18±44.77	34.39±33.03	<0.0001
C9	94.67±6.00	59.84±6.84	<0.0001
C10	90.45±14.48	69.15±17.63	<0.0001
C11	55.77±37.82	41.13±27.80	<0.0001

All of the data are reported as the sample *means* ± *SD*s. The locations of the TMS regions are shown as annotated by the `Swiss-Prot` database. There are statistically significant (P-value <0.0001) informative positions in the TMS regions compared to the non-TMS regions in the sequences from all classes except for the *nonselective* class, where the difference is not significant.

Table 34: Statistical analysis of the informative position rates close to TMS regions and far from TMS regions

Class ID	**Close**	**Far**	**P-value**
C1	78.24±23.09	53.31±26.22	0.002
C2	76.58±10.97	38.59±15.94	<0.0001
C3	66.82±26.47	43.77±22.95	<0.0001
C4	67.26±26.48	47.89±26.31	<0.0001
C5	78.15±19.54	50.94±21.79	<0.0001
C6	69.96±14.50	45.09±19.63	<0.0001
C7	69.18±17.71	43.39±20.65	<0.0001
C8	38.10±41.33	30.53±30.93	0.001
C9	76.60±06.79	49.55±11.43	<0.0001
C10	80.52±14.54	58.05±23.81	<0.0001
C11	49.91±33.30	34.75±26.89	<0.0001

All of the data are reported as the sample *means* ± *SD*s. For the non-TMS regions, there are statistically significant (P-value <0.0001) informative positions that occur close to the TMS regions (within 10 positions of the TMS) compared to other regions far from TMS regions in the sequences that belong to most classes, except the C1 (*nonselective*) and C8 (*Other organonitrogens*) classes, where the differences are not significant.

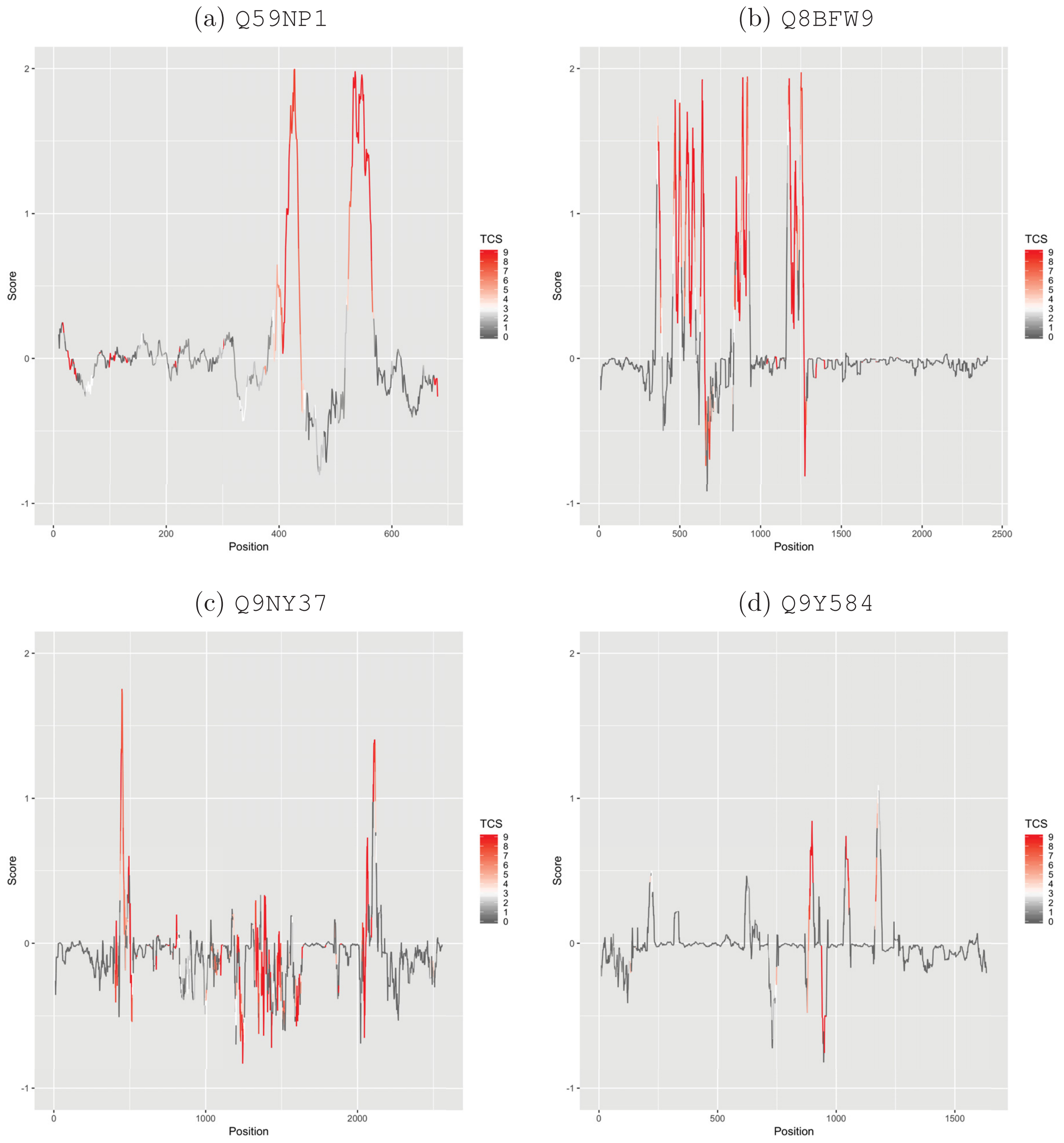


Figure 23: Average Kyte-Doolittle hydropathy of the MSAs with TCSs.

The figure indicates that the columns highlighted in red are informative and used by *TooT-SC*. The *TooT-SC* considers a column to be informative if it has a TCS of at least 4 (shades of red) and filters out the other columns (gray and white). In (a), `Q59NP1` contains 251 residues, and the alignment of `Q59NP1` with other homologous sequences has 692 columns; only 151 of them are informative (highlighted in shades of red). In (b), `Q8BFW9` contains 622 residues, and the alignment of `Q8BFW9` with other homologous sequences has 2,414 columns; only 439 of them are informative. In (c), `Q9NY37` contains 505 residues, and the alignment of `Q9NY37` with other homologous sequences has 2,568 columns; only 508 of them are informative. In (d), `Q9Y584` contains 194 residues, and the alignment of `Q9Y584` with other homologous sequences has 1,644 columns; only 79 of them are informative.

Table 35: Statistical analysis of the informative position rates in the interior and exterior TMS regions

Class ID	Interior	Exterior	P-value
C1	80.66±24.30	80.21±23.55	0.6485
C2	98.44±07.03	94.92±10.54	0.0003
C3	80.92±28.99	77.48±28.05	<0.0001
C4	81.74±28.33	79.10±27.18	<0.0001
C5	90.09±19.91	88.25±20.49	0.0001
C6	87.65±17.15	84.68±15.50	<0.0001
C7	83.93±17.22	81.31±16.97	<0.0001
C8	47.03±45.76	45.86±44.65	0.0641
C9	97.82±4.81	93.32±6.95	0.0001
C10	92.73±14.89	89.75±14.52	0.0002
C11	56.88±39.33	55.45±37.52	0.03335

All of the data are reported as the sample *means* ± *SD*s. For the TMS regions, there is no difference between the informative positions in the central one-third of the TMS regions and the remaining exterior regions in the sequences that belong to the C1 (*nonselective*), C2 (*water*), C5 (*organic anions*), C8 (*other organonitrogens*), C9 (*nucleotides*), C10 (*organic heterocyclics*), and C11 (*miscellaneous*) classes. The difference is significant in the sequences that belong to the C3 (*inorganic cations*), C4 (*inorganic anions*), C6 (*organo-oxygens*), and C7 (*amino acids and derivatives*) classes.

4 Conclusion

We have developed a novel method (*TooT-SC*) for the *de novo* prediction of substrates for membrane transporter proteins that combines information based on the amino acid composition, evolutionary information, and positional information. *TooT-SC* is able to efficiently classify transport proteins into eleven classes according to their transported substrate (i.e., *nonselective*, *water*, *inorganic cations*, *inorganic anions*, *organic anions*, *organo-oxygens*, *amino acids and derivatives*, *other organonitrogens*, *nucleotides*, *organic heterocyclics*, and *miscellaneous*); to the best of our knowledge, this is the highest number of classes offered by a *de novo* prediction tool. The *TooT-SC* method first incorporates the use of evolutionary information by taking 120 similar sequences and constructing an MSA using TM-Coffee. Next, it uses the positional information by filtering out unreliable positions, as determined by the TCS, and then uses the PAAC. The *TooT-SC* method achieved an overall MCC of 0.82 on an independent testing set, which is a 26% improvement over the state-of-the-art method. In addition, we evaluated the impact of each factor on the performance by incorporating evolutionary information and filtering out unreliable positions. We observed that the PAAC encoding outperforms other combinational variations. However, it does not show compelling performance on its own; the enhanced performance comes mainly from incorporating evolutionary and positional information.

Analysis of the location of the informative positions reveals that there are more statistically significant informative positions in the TMSs compared to the non-TMSs and there are more statistically significant informative positions that occur close to the TMSs compared to regions far from them. These findings provide a potential direction for future research to focus on these regions when incorporating evolutionary information.

Chapter 8

Conclusion

Numerous genome projects have resulted in a wealth of protein sequences, many of which are still unannotated. Membrane proteins are one of the least characterized proteins in terms of their structure and function due to their hydrophobic surfaces and poor conformational stability. Membrane proteins perform virtually all membrane functions, aside from the basic barrier property of the lipid bilayer. Transporters serve as gatekeepers that control the flow of molecules into and out of the cell and are attractive targets for the pharmaceutical industry. The main objectives of this research are solutions that correspond to major challenges in the annotation of proteins. The first objective is to detect membrane proteins. Detecting all types of membrane proteins is often overlooked when using transmembrane topology prediction to find TMSs, and when any are detected, it is assumed that the proteins of interest are membrane proteins. The experimental results suggest that this approach helps eliminate false-negative assignments, i.e., membranes assigned to nonmembranes, but fails to detect over 25% of membrane proteins. On the other hand, machine learning models trained using all types of membrane proteins help to mitigate this issue. Our results also suggest that encoding a protein based on merely its sequence does not show high discriminatory power compared with encodings that incorporate evolutionary information. This finding could be because while the protein sequence contains a great deal of important information, it also contains noise, and the use of evolutionary information places more emphasis on conserved residues likely to have functional roles, thus helping to improve the discriminatory power. Finally, combining both transmembrane topology prediction and the predictions from machine learning classifiers exploits the strengths of both approaches and boosts the overall accuracy of membrane protein detection. The proposed integrative approach *TooT-M* combines both transmembrane topology prediction and predictions from machine learning classifiers; the comparative performance results indicate that *TooT-M* outperforms the state-of-the-art methods in terms of accuracy and MCC.

The second objective is to distinguish transporter proteins from non-transporter proteins. To accomplish this aim, there are two typical approaches: annotation transfer by homology and traditional machine learning methods. Our results suggest that annotation transfer by homology requires a trade-off between sensitivity and specificity. A stricter threshold eliminates false-negative assignments but at the cost of lower true-positive assignments, while a less strict threshold increases the true-positive assignments and the false-negative assignments. A good balance is achieved by the thresholds suggested by Aplop and Butler [AB17]. Nevertheless, annotation transfer by homology achieves a lower overall MCC than machine learning models that utilize features with evolutionary information. Interestingly, the predictions from annotation transfer by homology and the predictions from traditional machine learning methods have a lower correlation, which makes them good candidates for an ensemble classifier. The *TooT-T* ensemble classifier proposed by this research is trained to optimally combine the predictions from annotation transfer by homology and traditional machine learning models. The experimental results indicate that *TooT-T* achieves a performance superior to all the other approaches, outperforming all of the state-of-the-art *de novo* prediction methods in terms of accuracy and MCC.

The third objective is to facilitate the data collection process for the substrate specificity of transporters. The ultimate goal of transporter substrate specificity methods is to predict the exact substrate(s) that a transporter transports across the membrane; the analysis of the number of annotated transporters reveals that this goal is not feasible with the current number of annotated transporters. This lack of data justifies why substrate prediction researchers predict the general class to which the substrate belongs (e.g., amino acids) instead of the exact substrate(s) (e.g., arginine). However, the assignment of a general class in place of exact substrates is not explained or documented, which makes replicating or expanding a dataset to account for newly annotated transporters extremely difficult and subject to errors and inconsistencies. The proposed tool (*Ontoclass*) automates the data collection process for transporter substrate-specific datasets. *Ontoclass* applies Saier's classification system for substrates and relies on gold standard ontologies and databases to make the assignment. *Ontoclass* makes it possible to construct and expand a dataset in a consistent, explainable, and reproducible manner.

The fourth objective is to broaden the scope of substrate class prediction while still maintaining a credible predictive performance. The largest number of substrates predicted by the state-of-the-art methods is seven substrate classes. To increase this number, we utilized *Ontoclass* to construct a benchmark transporter dataset from the `Swiss-Prot` database and modified it so that the classes with a very small number of samples are grouped into a higher level of abstraction. The final dataset contains eleven different substrate classes: *nonselective*, *water*, *inorganic cations*, *inorganic anions*, *organic anions*, *organo-oxygens*, *amino acids and derivatives*, *other organonitrogens*, *nucleotides*, *organic heterocyclics*, and *miscellaneous.* Using this dataset, we developed a novel tool, *TooT-SC*, that utilizes compositional, evolutionary and positional information. The experimental results suggest that this method outperforms all of the state-of-the-art methods in terms of the overall accuracy and MCC. Furthermore, the analysis of the reliable positions in the alignments reveals that the TMSs contain more statistically significant informative positions than the non-TMSs, and more statistically significant informative positions occur close to the TMSs than in regions far away from them. This finding offers a direction for future work to target these regions, as more important information seems to be located there.

1 Contributions

The main contributions of this thesis are summarized below:

1.1 Improving computational approaches to detect membrane proteins

We have demonstrated the limitation of merely using transmembrane topology prediction to detect all types of membrane proteins. The performances of different feature extraction techniques, including those employed by state-of-the-art predictors, were examined with the combination of the KNN, OET-KNN, SVM, and GBM machine learning algorithms. The experimental results suggest that features that incorporate evolutionary information outperform features that rely on a single protein sequence. In addition, an ensemble classifier that combines the results from a selected set of Pse-PSSM OET-KNN predictors outperforms all of the other approaches. The selected set was chosen so that it would have low within-set correlation and high correlation with the target class. This is an improved version of the MemType-2L technique that reduces the number of constituent classifiers by over 90% while enhancing the performance. The proposed approach (*TooT-M*) combines the predictions obtained from the selected set of classifiers and the prediction from the TOPCONS2 transmembrane topology prediction tool. Experiments on multiple datasets suggest that *TooT-M* outperforms all of the state-of-the-art methods in terms of accuracy and MCC.

1.2 Improving computational approaches to detect transporter proteins

We proposed a new way to encode a protein sequence by combining traditional compositions with evolutionary information called *psi-composition*. The combination of AAC with evolutionary information from a BLAST output has been performed previously in the literature [SCH10]. To the best of our knowledge, the combination of PAAC and Pse-AAC with evolutionary information is a novel contribution of this research. The results suggest that among all of the tested encodings, the psiPAAC encoding, which incorporates PAAC with PSI-BLAST output, achieves the highest discriminatory power.

The proposed tool for transporter detection (*TooT-T*) is trained to optimally combine the predictions from homology annotation transfer and machine learning methods to determine the final prediction. Homology annotation transfer detects transporters by searching against the `TCDB` under three different thresholds. The machine learning methods include three SVM models wherein the protein sequences are encoded using the proposed *psi-composition* encodings. The experimental results obtained by cross-validation and independent testing show that the combination of the two approaches (i.e., homology annotation transfer and machine learning methods) is more beneficial than employing only one method. Further, *TooT-T* outperforms all of the state-of-the-art methods that rely on the protein sequence alone in terms of accuracy and MCC.

1.3 Facilitating the substrate-specific data collection process

An obstacle encountered when establishing a substrate-specific transporter dataset is that the number of annotated transporters with specific substrates is limited. This limitation led researchers to assign a general label to more specific substrates to produce a dataset with a sufficient number of samples in each class. This assignment shift from specific substrates to a general class is not documented or explained, which makes reproducing the same dataset or expanding it to include newly annotated transporters both time consuming and subject to inconsistencies.

The proposed tool (*Ontoclass*) automates the data collection process for transporter substrate-specific datasets. *Ontoclass* determines if a protein has transporter-related `GO` MF annotations in the `Swiss-Prot` database and obtains the `ChEBI`-IDs of the transported substrates from the `GO` annotation. These `ChEBI`-IDs and their ancestors are used to identify the class according to Saier's classification system. The tool outputs the final substrate class of each protein along with additional information. The automated tool relieves the burden of manual curation, is consistent with other ontologies, and is reproducible. In addition, it can adapt to the exponential growth of and updates to biological databases with minimal prior knowledge.

1.4 Broadening the scope of substrate-specific prediction

The proposed tool (*TooT-SC*) is able to efficiently classify transport proteins according to eleven substrate classes: *nonselective*, *water*, *inorganic cations*, *inorganic anions*, *organic anions*, *organo-oxygens*, *amino acids and derivatives*, *other organonitrogens*, *nucleotides*, *organic heterocyclics*, and *miscellaneous*. This is the highest number of classes predicted by any transporter substrate specificity detection tool.

In addition, *TooT-SC* utilizes the PAAC encoding scheme, the TM-Coffee MSA algorithm [CDTTN12] and the TCS algorithm [CDTN14] to determine informative positions in the MSA. The use of evolutionary information in the form of an MSA on the composition-encoding PAAC shows a considerable positive impact in most substrate classes, where the average improvement in the MCC is 126.41%. The further use of positional information by filtering out unreliable columns from the MSA shows an average improvement of 128.57% over the baseline PAAC. The impact of positional information over that already achieved by evolutionary information is positive in most substrate classes. The combination of the three sources of information is a novel contribution of this research. Our results suggest that the *TooT-SC* method outperforms all of the state-of-the-art methods in terms of the overall accuracy and MCC.

Furthermore, the analysis of informative positions in the alignments reveals that there are more statistically significant informative positions located in the TMSs than in the non-TMSs, as well as more statistically significant informative positions close to the TMSs compared to regions far away from them.

2 Data availability

The datasets used in this research are available online at:
`https://tootsuite.encs.concordia.ca/datasets/`

The proposed tools are available at:
`https://github.com/bioinformatics-group/`

3 Limitations and future directions

A major focus of this research is on feature engineering, (i.e., encoding the protein sequence in a numerical vector in a way that encapsulates it function); from our exterminations, we learned that evolutionary information is the driving force behind high performance. In all of the experiments and in all of the tasks, features with evolutionary information outperformed those extracted from a single protein sequence. One of desirable aims is to automatically *learn* features from the data and to discover the representations needed to build a classification from raw sequences. Word embeddings are a feature learning technique in NLP, where words are represented in dense n-dimensional vectors in a way that preserves their semantic relationships. Many studies have proposed treating the protein sequence as a "sentence" of "words" and applying techniques similar to those in NLP to learn the word-embedding representation of a protein such as in [AM15,YWBA18,HPO+19]. Our initial attempts to do so were not as successful as those using features with evolutionary information; this could be because one fundamental difference between protein sequences and natural language is that the words of similar meaning rely on evolutionary information rather than the word itself; two words could have completely different relationships depending of the evolutionary context of a given protein. How can we incorporate such evolutionary information into word embedding? A straightforward approach is to manually incorporate the evolutionary information to learned vectors from word embedding; other possible approaches include applying deep learning to extract fixed-length features from the sequence-length-dependent features, as done by [LWU+18]. We see potential in these very interesting directions for future work.

Further, the solution proposed for each problem is a working solution, and the results are promising; however, the tools were not applied in the annotation of a complete genome. Applying the tools to annotate a complete genome followed by an analysis of the findings are initial future directions.

For substrate specificity prediction, although the proposed tool *TooT-SC* predicts eleven substrate classes, which is the highest number of substrate classes offered by any *de novo* prediction tool, it still falls short of the ultimate goal of predicting the exact substrate(s). This goal is mainly obstructed by the limited number of annotated transporters. Thus, a periodic investigation and the expansion of the predicted substrates as sufficient experimental data becomes available is necessary. In addition, all substrate specificity detection methods, including *TooT-SC*, overlook that the relationship between the transporter and the substrate is not one-to-one. For example, a transporter could transport more than one type of substrate. Granted, dealing with overlapping classes while single class methods are yet far from being established is illogical. However, experiments on multiclass transporters and substrates are a desirable direction for the future research.

The framework of the *TooT-SC* tool combines PAAC, the TM-Coffee MSA algorithm, and the TCS [CDTN14] algorithm for determining informative positions in the MSA. The results indicate that the combination of the three sources (i.e., compositional, evolutionary and positional information) is advantageous. However, the computational time is not optimal for a genome-scale annotation, as it takes an average of 15 minutes to compute the encoding vector of a given protein on a MacBook Pro with an Intel Core i7 @ 2.9 GHz processor (16 GB 1 2133 MHz LPDDR3 and 1 TB HD storage). The majority (95%) of the computational cost comes from the TCS filtering step. Thus, exploring efficient alternatives is worthwhile.

Bibliography

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

[AB17] Faizah Aplop and Greg Butler. TransATH: Transporter prediction via annotation transfer by homology. *ARPN Journal of Engineering and Applied Sciences*, 12(2), 2017.

[AB19] Munira Alballa and Gregory Butler. Ontology-based transporter substrate annotation for benchmark datasets. In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2613–2619. IEEE, 2019.

[AB20] Munira Alballa and Gregory Butler. TooT-T: discrimination of transport proteins from non-transport proteins. *BMC Bioinformatics*, 21:1–10, 2020.

[ABB$^+$00] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25, 2000.

[ABW$^+$04] Rolf Apweiler, Amos Bairoch, Cathy H Wu, Winona C Barker, Brigitte Boeckmann, Serenella Ferro, Elisabeth Gasteiger, Hongzhan Huang, Rodrigo Lopez, Michele Magrane, et al. UniProt: the universal protein knowledgebase. *Nucleic Acids Research*, 32(suppl_1):D115–D119, 2004.

[Agg14] Charu C Aggarwal. *Data Classification: Algorithms and Applications.* CRC Press, 2014.

[AHJ18] Muhammad Arif, Maqsood Hayat, and Zahoor Jan. iMem-2LSAAC: A two-level model for discrimination of membrane proteins and their types by extending the notion of SAAC into Chou's pseudo amino acid composition. *Journal of Theoretical Biology*, 442:11–21, 2018.

[AM15] Ehsaneddin Asgari and Mohammad RK Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PLoS ONE*, 10(11):e0141287, 2015.

[AMS$^+$97] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[Apl16] Faizah Aplop. *Computational Approaches To Improving The Reconstruction Of Metabolic Pathway.* PhD thesis, Concordia University, 2016.

[BBA$^+$03] Brigitte Boeckmann, Amos Bairoch, Rolf Apweiler, Marie-Claude Blatter, Anne Estreicher, Elisabeth Gasteiger, Maria J Martin, Karine Michoud, Claire O'Donovan, Isabelle Phan, et al. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research*, 31(1):365–370, 2003.

[BDA13] Mohamed Bekkar, Hassiba Kheliouane Djemaa, and Taklit Akrouf Alitouche. Evaluation measures for models assessment over imbalanced data sets. *Journal Of Information Engineering and Applications*, 3(10), 2013.

[BE$^+$94] Timothy L Bailey, Charles Elkan, et al. Fitting a mixture model by expectation maximization to discover motifs in bipolymers. 1994.

[BFJE04] Frode S Berven, Kristian Flikka, Harald B Jensen, and Ingvar Eidhammer. BOMP: a program to predict integral β-barrel outer membrane proteins encoded within genomes of Gram-negative bacteria. *Nucleic Acids Research*, 32:W394–W399, 2004.

[BGA$^+$17] Basharat Bhat, Nazir A Ganai, Syed Mudasir Andrabi, Riaz A Shah, and Ashutosh Singh. TM-Aligner: Multiple sequence alignment tool for transmembrane proteins with reduced time and improved accuracy. *Scientific Reports*, 7(1):12543, 2017.

[BGJM17] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[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.

[BKJ$^+$16] Ahmad Hassan Butt, Sher Afzal Khan, Hamza Jamil, Nouman Rasool, and Yaser Daanial Khan. A prediction model for membrane proteins using moments based features. *BioMed Research International*, 2016, 2016.

[BLH05] Pantelis G Bagos, Theodore D Liakopoulos, and Stavros J Hamodrakas. Evaluation of methods for predicting the topology of β-barrel outer membrane proteins and a consensus prediction method. *BMC Bioinformatics*, 6(1):7, 2005.

[BLSH04] Pantelis G Bagos, Theodore D Liakopoulos, Ioannis C Spyropoulos, and Stavros J Hamodrakas. PRED-TMBB: a web server for predicting the topology of β-barrel outer membrane proteins. *Nucleic Acids Research*, 32(suppl_2):W400–W404, 2004.

[BRK17] Ahmad Hassan Butt, Nouman Rasool, and Yaser Daanial Khan. A treatise to computational approaches towards prediction of membrane protein and its subtypes. *The Journal of Membrane Biology*, 250(1):55–76, 2017.

[BSS$^+$10] Gordon Blackshields, Fabian Sievers, Weifeng Shi, Andreas Wilm, and Desmond G Higgins. Research sequence embedding for fast construction of guide trees for multiple sequence alignment. *Algorithms for Molecular Biology*, 5:21, 2010.

[Bue15] Lukas Buehler. The Structure of Membrane Proteins. In *Cell Membranes*, chapter 3. Garland Science, 2015.

[BVF$^+$08] Andreas Bernsel, Håkan Viklund, Jenny Falk, Erik Lindahl, Gunnar von Heijne, and Arne Elofsson. Prediction of membrane-protein topology from first principles. *Proceedings of the National Academy of Sciences*, 105(20):7177–7181, 2008.

[CAPPQ97] Juan Cedano, Patrick Aloy, Josep A Perez-Pons, and Enrique Querol. Relation between amino acid composition and cellular location of proteins. *Journal of Molecular Biology*, 266(3):594–600, 1997.

[CBCI08] Elisabeth P Carpenter, Konstantinos Beis, Alexander D Cameron, and So Iwata. Overcoming the challenges of membrane protein crystallography. *Current Opinion in Structural Biology*, 18(5):581–586, 2008.

[CC14] Abhijit Chakraborty and Saikat Chakrabarti. A survey on prediction of specificity-determining sites in proteins. *Briefings in Bioinformatics*, 16(1):71–88, 2014.

[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.

[CDTTN12] Jia-Ming Chang, Paolo Di Tommaso, Jean-François Taly, and Cedric Notredame. Accurate multiple sequence alignment of transmembrane proteins with PSI-Coffee. *BMC Bioinformatics*, 13(Suppl 4):S1, 2012.

[Cho01] Kuo-Chen Chou. Prediction of protein cellular attributes using pseudo-amino acid composition. *Proteins: Structure, Function, and Bioinformatics*, 43(3):246–255, 2001.

[CMB$^+$14] Marcus C Chibucos, Christopher J Mungall, Rama Balakrishnan, Karen R Christie, Rachael P Huntley, Owen White, Judith A Blake, Suzanna E Lewis, and Michelle Giglio. Standardized description of scientific evidence using the Evidence Ontology (ECO). *Database*, 2014, 2014.

[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.

[CS07] Kuo-Chen Chou and Hong-Bin Shen. MemType-2L: a web server for predicting membrane proteins and their types by incorporating evolution information through Pse-PSSM. *Biochemical and Biophysical Research Communications*, 360(2):339–345, 2007.

[CZ95] Kuo-Chen Chou and Chun-Ting Zhang. Prediction of protein structural classes. *Critical Reviews in Biochemistry and Molecular Biology*, 30(4):275–349, 1995.

[Den95] Thierry Denoeux. A K-nearest neighbor classification rule based on Dempster-Shafer theory. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(5):804–813, 1995.

[Din11] Zejin Ding. *Diversified ensemble classifiers for highly imbalanced data learning and its application in bioinformatics*. PhD thesis, Georgia State University, 2011.

[DOS13] Jurate Daugelaite, Aisling O'Driscoll, and Roy D Sleator. An overview of multiple sequence alignments and cloud computing in bioinformatics. *ISRN Biomathematics*, 2013, 2013.

[DRFR15] Oscar Dias, Miguel Rocha, Eugénio C Ferreira, and Isabel Rocha. Reconstructing genome-scale metabolic models with merlin. *Nucleic Acids Research*, 43(8):3899–3910, 2015.

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

[FBS$^+$14] Alexander Farwick, Stefan Bruder, Virginia Schadeweg, Mislav Oreb, and Eckhard Boles. Engineering of yeast hexose transporters to transport D-xylose without inhibition by D-glucose. *Proceedings of the National Academy of Sciences*, 111(14):5159–5164, 2014.

[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.

[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.

[G$^+$06] Louis Sanford Goodman et al. *Goodman and Gilman's the pharmacological basis of therapeutics*. McGraw-Hill New York, 11 edition, 2006.

[GAW05] Andrew G Garrow, Alison Agnew, and David R Westhead. TMB-Hunt: an amino acid composition based method to screen proteomes for beta-barrel transmembrane proteins. *BMC Bioinformatics*, 6(1):56, 2005.

[GNY$^+$13] Ilya Getsin, Gina H Nalbandian, Daniel C Yee, Ake Vastermark, Philipp CG Paparoditis, Vamsee S Reddy, and Milton H Saier. Comparative genomics of transport proteins in developmental bacteria: Myxococcus xanthus and streptomyces coelicolor. *BMC Microbiology*, 13(1):279, 2013.

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

[Got96] Osamu Gotoh. Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. *Journal of Molecular Biology*, 264(4):823–838, 1996.

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

[HAB$^+$13] David P Hill, Nico Adams, Mike Bada, Colin Batchelor, Tanya Z Berardini, Heiko Dietze, Harold J Drabkin, Marcus Ennis, Rebecca E Foulger, Midori A Harris, et al. Dovetailing biology and chemistry: integrating the Gene Ontology with the ChEBI chemical ontology. *BMC Genomics*, 14(1):513, 2013.

[Hay12] Hayat, Maqsood and Khan, Asifullah. MemHyb: predicting membrane protein types by hybridizing SAAC and PSSM. *Journal of Theoretical Biology*, 292:93–102, 2012.

[HE12] Sikander Hayat and Arne Elofsson. BOCTOPUS: improved topology prediction of transmembrane β barrel proteins. *Bioinformatics*, 28(4):516–522, 2012.

[HGS$^+$15] Yayun Hu, Yanzhi Guo, Yinan Shi, Menglong Li, and Xuemei Pu. A consensus subunit-specific model for annotation of substrate specificity for ABC transporters. *RSC Advances*, 5(52):42009–42019, 2015.

[HOD$^+$15] Janna Hastings, Gareth Owen, Adriano Dekker, Marcus Ennis, Namrata Kale, Venkatesh Muthukrishnan, Steve Turner, Neil Swainston, Pedro Mendes, and Christoph Steinbeck. ChEBI in 2016: Improved services and an expanding collection of metabolites. *Nucleic Acids Research*, 44(D1):D1214–D1219, 2015.

[HPO$^+$19] Quang-Thai Ho, Dinh-Van Phan, Yu-Yen Ou, et al. Using word embedding technique to efficiently represent protein sequences for identifying substrate specificities of transporters. *Analytical Biochemistry*, 577:73–81, 2019.

[HS15] K Harini and Ramanathan Sowdhamini. Computational approaches for decoding select odorant-olfactory receptor interactions using mini-virtual screening. *PLoS ONE*, 10(7), 2015.

[HSW$^+$10] Tao Huang, Xiao-He Shi, Ping Wang, Zhisong He, Kai-Yan Feng, LeLe Hu, Xiangyin Kong, Yi-Xue Li, Yu-Dong Cai, and Kuo-Chen Chou. Analysis and prediction of the metabolic stability of proteins based on their sequential features, subcellular locations and interaction networks. *PLoS ONE*, 5(6), 2010.

[HW81] Thomas P Hopp and Kenneth R Woods. Prediction of protein antigenic determinants from amino acid sequences. *Proceedings of the National Academy of Sciences*, 78(6):3824–3828, 1981.

[HY08] Jing Hu and Changhui Yan. A method for discovering transmembrane β-barrel proteins in gram-negative bacterial proteomes. *Computational Biology and Chemistry*, 32(4):298–301, 2008.

[IKY02] Takeshi Inoue, Ichiro Kusumi, and Mitsuhiro Yoshioka. Serotonin transporters. *Current Drug Targets-CNS & Neurological Disorders*, 1(5):519–529, 2002.

[JMF$^+$01] Irene Jacoboni, Pier Luigi Martelli, Piero Fariselli, Vito De Pinto, and Rita Casadio. Prediction of the transmembrane regions of β-barrel membrane proteins with a neural network-based predictor. *Protein Science*, 10(4):779–787, 2001.

[KD82] Jack Kyte and Russell F Doolittle. A simple method for displaying the hydropathic character of a protein. *Journal of Molecular Biology*, 157(1):105–132, 1982.

[KK00] Shuichi Kawashima and Minoru Kanehisa. AAindex: amino acid index database. *Nucleic Acids Research*, 28(1):374–374, 2000.

[KKS05] Lukas Käll, Anders Krogh, and Erik LL Sonnhammer. An HMM posterior decoder for sequence feature prediction that includes homology information. *Bioinformatics*, 21(suppl_1):i251–i257, 2005.

[KLvHS01] Anders Krogh, BjoÈrn Larsson, Gunnar von Heijne, and Erik LL Sonnhammer. Predicting transmembrane protein topology with a hidden markov model: application to complete genomes. *Journal of Molecular Biology*, 305(3):567–580, 2001.

[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.

[Koh95] Ron Kohavi. Wrappers for performance enhancement and oblivious decision graphs. Technical report, Carnegie-Mellon University, 1995.

[KPP$^+$07] Shuichi Kawashima, Piotr Pokarowski, Maria Pokarowska, Andrzej Kolinski, Toshiaki Katayama, and Minoru Kanehisa. AAindex: amino acid index database, progress report 2008. *Nucleic Acids Research*, 36:D202–D205, 2007.

[Kri07] Evgeny Krissinel. On the relationship between sequence and structure similarities in proteomics. *Bioinformatics*, 23(6):717–723, 2007.

[KT08] Kazutaka Katoh and Hiroyuki Toh. Recent developments in the MAFFT multiple sequence alignment program. *Briefings in Bioinformatics*, 9(4):286–298, 2008.

[KV95] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems*, pages 231–238, 1995.

[KYB03] Ian Korf, Mark Yandell, and Joseph Bedell. *BLAST*, chapter 4, pages 57–60. O'Reilly Media, Inc., 2003.

[LBUZ09] Haiquan Li, Vagner A Benedito, Michael K Udvardi, and Patrick X Zhao. TransportTP: a two-phase classification approach for membrane transporter prediction and characterization. *BMC Bioinformatics*, 10(1):418, 2009.

[LBZ$^+$00] Harvey Lodish, Arnold Berk, Lawrence Zipursky, Paul Matsudaira, David Baltimore, and James Darnell. Transport across cell membranes. In *Molecular Cell Biology*, chapter 15. W. H. Freeman, New York, 2000.

[LG06] Weizhong Li and Adam Godzik. CD-HIT: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006.

[Lin08] Hao Lin. The modified Mahalanobis discriminant for predicting outer membrane proteins by using Chou's pseudo amino acid composition. *Journal of Theoretical Biology*, 252(2):350–356, 2008.

[LLJ$^+$11] Feiran Lu, Shuo Li, Yang Jiang, Jing Jiang, He Fan, Guifeng Lu, Dong Deng, Shangyu Dang, Xu Zhang, Jiawei Wang, et al. Structure and mechanism of the uracil transporter UraA. *Nature*, 472(7342):243–246, 2011.

[LLX$^+$16] Liqi Li, Jinhui Li, Weidong Xiao, Yongsheng Li, Yufang Qin, Shiwen Zhou, and Hua Yang. Prediction the substrate specificities of membrane transport proteins based on support vector machine and hybrid features. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 13(5):947–953, 2016.

[LSM10] Yongchao Liu, Bertil Schmidt, and Douglas L Maskell. MSAProbs: multiple sequence alignment based on pair hidden markov models and partition function posterior probabilities. *Bioinformatics*, 26(16):1958–1964, 2010.

[LVY$^+$15] Yi-Fan Liou, Tamara Vasylenko, Chia-Lun Yeh, Wei-Chun Lin, Shih-Hsiang Chiu, Phasit Charoenkwan, Li-Sun Shu, Shinn-Ying Ho, and Hui-Ling Huang. SCMMTP: identifying and characterizing membrane transport proteins using propensity scores of dipeptides. *BMC Genomics*, 16(12):S6, 2015.

[LWU$^+$18] Yu Li, Sheng Wang, Ramzan Umarov, Bingqing Xie, Ming Fan, Lihua Li, and Xin Gao. DEEPre: sequence-based enzyme EC number prediction by deep learning. *Bioinformatics*, 34(5):760–769, 2018.

[LZS15] Nicolas Loira, Anna Zhukova, and David James Sherman. Pantograph: A template-based method for genome-scale metabolic model reconstruction. *Journal of Bioinformatics and Computational Biology*, 13(02):1550006, 2015.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[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):e100278, 2014.

[MRS08] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge university press, 2008.

[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.

[NJ09] Timothy Nugent and David T Jones. Transmembrane protein topology prediction using support vector machines. *BMC Bioinformatics*, 10:159(1), 2009.

[NKT83] Ken Nishikawa, Yasushi Kubota, and OOI Tatsuo. Classification of proteins into groups based on amino acid composition and other characters. I. Angular distribution. *Journal of Biochemistry*, 94(3):981–995, 1983.

[NNT86] Hiroshi Nakashima, Ken Nishikawa, and OOI Tatsuo. The folding type of a protein is relevant to the amino acid composition. *Journal of Biochemistry*, 99(1):153–162, 1986.

[OAF10] Clare M O'Connor, Jill U Adams, and Jennifer Fairman. Essentials of cell biology. *Cambridge, MA: NPG Education*, 1, 2010.

[OCG10] Yu-yen Ou, Shu-an Chen, and M Michael Gromiha. Prediction of membrane spanning segments and topology in β-barrel membrane proteins at better accuracy. *Journal of Computational Chemistry*, 31(1):217–223, 2010.

[OGCS08] YuYen Ou, M Michael Gromiha, ShuAn Chen, and Makiko Suwa. TMBETADISC-RBF: discrimination of β-barrel membrane proteins using RBF networks and PSSM profiles. *Computational Biology and Chemistry*, 32(3):227–231, 2008.

[OS96] David W Opitz and Jude W Shavlik. Generating accurate and diverse members of a neural-network ensemble. In *Advances in Neural Information Processing Systems*, pages 535–541, 1996.

[PdROC14] Fabiano Sviatopolk-Mirsky Pais, Patrícia de Ruy, Guilherme Oliveira, and Roney Coimbra. Assessing the efficiency of multiple sequence alignment programs. *Algorithms for Molecular Biology*, 9(4), 2014.

[Pea13] William R Pearson. An introduction to sequence similarity ("homology") searching. *Current Protocols in Bioinformatics*, 42(1):3–1, 2013.

[Pev09] Jonathan Pevsner. Multiple Sequence Alignment. In *Bioinformatics and Functional Genomics*, chapter 6. Wiley-Blackwell, Baltimore , Maryland, 2 edition, 2009.

[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.

[PLD05] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[PVL$^+$14] Philipp Paparoditis, Åke Västermark, Andrew J Le, John A Fuerst, and Milton H Saier Jr. Bioinformatic analyses of integral membrane transport proteins encoded within the genome of the planctomycetes species, rhodopirellula baltica. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 1838(1):193–215, 2014.

[RKR$^+$08] Sheila M Reynolds, Lukas Käll, Michael E Riffle, Jeff A Bilmes, and William Stafford Noble. Transmembrane topology and signal peptide prediction using dynamic bayesian networks. *PLoS Computational Biology*, 4(11):e1000213, 2008.

[RLLS09] Michael Remmert, Dirk Linke, Andrei N Lupas, and Johannes Söding. HHomppprediction and classification of outer membrane proteins. *Nucleic Acids Research*, 37(suppl_2):W446–W451, 2009.

[RP05] Qinghu Ren and Ian T Paulsen. Comparative analyses of fundamental differences in membrane transport capabilities in prokaryotes and eukaryotes. *PLoS Computational Biology*, 1(3):e27, 2005.

[Sai00] Milton H Saier. A functional-phylogenetic classification system for transmembrane solute transporters. *Microbiology and Molecular Biology Reviews*, 64(2):354–411, 2000.

[SAJT14] Swagatika Sahoo, Maike Kathrin Aurich, Jon Johannes Jonsson, and Ines Thiele. Membrane transporters in a human genome-scale metabolic knowledgebase and their implications for disease. *Frontiers in Physiology*, 5:91, 2014.

[SC07] Hong-Bin Shen and Kuo-Chen Chou. EzyPred: a top–down approach for predicting enzyme functional classes and subclasses. *Biochemical and Biophysical Research Communications*, 364(1):53–59, 2007.

[Sch03] Georg E Schulz. Transmembrane β-barrel proteins. *Advances in Protein Chemistry*, 63:47–70, 2003.

[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.

[SG04] Yinon Shafrir and H Robert Guy. STAM: simple transmembrane alignment method. *Bioinformatics*, 20(5):758–769, 2004.

[SGW$^+$11] Nitesh Kumar Singh, Aaron Goodman, Peter Walter, Volkhard Helms, and Sikander Hayat. TMBHMM: a frequency profile based HMM for predicting the topology of transmembrane beta barrel proteins and the exposure status of transmembrane residues. *Biochimica et Biophysica Acta (BBA)-Proteins and Proteomics*, 1814(5):664–670, 2011.

[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$^+$15] 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, 2015.

[SJTB06] Milton H Saier Jr, Can V Tran, and Ravi D Barabote. TCDB: the Transporter Classification Database for membrane transport protein analyses and information. *Nucleic Acids Research*, 34(suppl_1):D181–D186, 2006.

[Söd05] Johannes Söding. Protein homology detection by HMM–HMM comparison. *Bioinformatics*, 21(7):951–960, 2005.

[SSS04] Yasushi Shigeri, Rebecca P Seal, and Keiko Shimamoto. Molecular pharmacology of glutamate transporters, eaats and vgluts. *Brain Research Reviews*, 45(3):250–265, 2004.

[SWD$^+$11] Fabian Sievers, Andreas Wilm, David Dineen, Toby J Gibson, Kevin Karplus, Weizhong Li, Rodrigo Lopez, Hamish McWilliam, Michael Remmert, Johannes

Söding, et al. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, 7:539(1), 2011.

[Tan62] Charles Tanford. Contribution of hydrophobic interactions to the stability of the globular conformation of proteins. *Journal of the American Chemical Society*, 84(22):4240–4247, 1962.

[TEB16] Konstantinos D Tsirigos, Arne Elofsson, and Pantelis G Bagos. PRED-TMBB2: improved topology prediction and detection of beta-barrel outer membrane proteins. *Bioinformatics*, 32(17):i665–i671, 2016.

[TGB$^+$18] Konstantinos D Tsirigos, Sudha Govindarajan, Claudio Bassot, Åke Västermark, John Lamb, Nanjiang Shu, and Arne Elofsson. Topology of membrane proteins-predictions, limitations and variations. *Current Opinion in Structural Biology*, 50:9–17, 2018.

[THG94] Julie D Thompson, Desmond G Higgins, and Toby 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.

[THKE12] Konstantinos D Tsirigos, Aron Hennerdal, Lukas Käll, and Arne Elofsson. A guideline to proteome-wide α-helical membrane protein topology predictions. *Proteomics*, 12(14):2282–2294, 2012.

[TLLP11] Julie D Thompson, Benjamin Linard, Odile Lecompte, and Olivier Poch. A comprehensive benchmark study of multiple sequence alignment methods: current challenges and future perspectives. *PLoS ONE*, 6(3):e18093, 2011.

[TP10] Ines Thiele and Bernhard Ø Palsson. A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nature Protocols*, 5(1):93, 2010.

[TPS$^+$15] Konstantinos D Tsirigos, Christoph Peters, Nanjiang Shu, Lukas Käll, and Arne Elofsson. The TOPCONS web server for consensus prediction of membrane protein topology and signal peptides. *Nucleic Acids Research*, 43(W1):W401–W407, 2015.

[TS98] Gabor E Tusnady and Istvan Simon. Principles governing amino acid composition of integral membrane proteins: application to topology prediction. *Journal of Molecular Biology*, 283(2):489–506, 1998.

[TS01] Gabor E Tusnady and Istvan Simon. The HMMTOP transmembrane topology prediction server. *Bioinformatics*, 17(9):849–850, 2001.

[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(1):235, 2012.

[VBSE08] Håkan Viklund, Andreas Bernsel, Marcin Skwark, and Arne Elofsson. SPOCTOPUS: a combined predictor of signal peptides and membrane protein topology. *Bioinformatics*, 24(24):2928–2929, 2008.

[VE08] Håkan Viklund and Arne Elofsson. OCTOPUS: improving topology prediction by two-track ANN-based preference scores and an extended topological grammar. *Bioinformatics*, 24(15):1662–1668, 2008.

[vH92] Gunnar von Heijne. Membrane hydrophobicity protein structure prediction analysis and the positive-inside. *Journal of Molecular Biology*, 225:487–94, 1992.

[vH99] Gunnar von Heijne. Recent advances in the understanding of membrane protein assembly and structure. *Quarterly Reviews of Biophysics*, 32(4):285–307, 1999.

[WL03] James C Whisstock and Arthur M Lesk. Prediction of protein function from protein sequence and structure. *Quarterly Reviews of Biophysics*, 36(03):307–340, 2003.

[Wol92] David H Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[WP03] Gary M Weiss and Foster Provost. Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–354, 2003.

[WvH98] Erik Wallin and Gunnar von Heijne. Genome-wide analysis of integral membrane proteins from eubacterial, archaean, and eukaryotic organisms. *Protein Science*, 7(4):1029–1038, 1998.

[XWC11] Xuan Xiao, Pu Wang, and Kuo-Chen Chou. GPCR-2L: Predicting G protein-coupled receptors and their types by hybridizing two different modes of pseudo amino acid compositions. *Molecular Biosystems*, 7(3):911–919, 2011.

[YSJ12] Jiwon Youm and Milton H Saier Jr. Comparative analyses of transport proteins encoded within the genomes of mycobacterium tuberculosis and mycobacterium leprae. *Biochimica et Biophysica Acta (BBA)-Biomembranes*, 1818(3):776–797, 2012.

[YWBA18] Kevin K Yang, Zachary Wu, Claire N Bedbrook, and Frances H Arnold. Learned protein embeddings for machine learning. *Bioinformatics*, 34(15):2642–2648, 2018.

Appendices

Appendix A

Detailed performance evaluation of Pse-PSSM where $\lambda \in (0, \ldots, 49)$

Figure 24 delineates the accuracy of different models trained using the Pse-PSSM encodings ($\lambda \in (0, \ldots, 49)$).

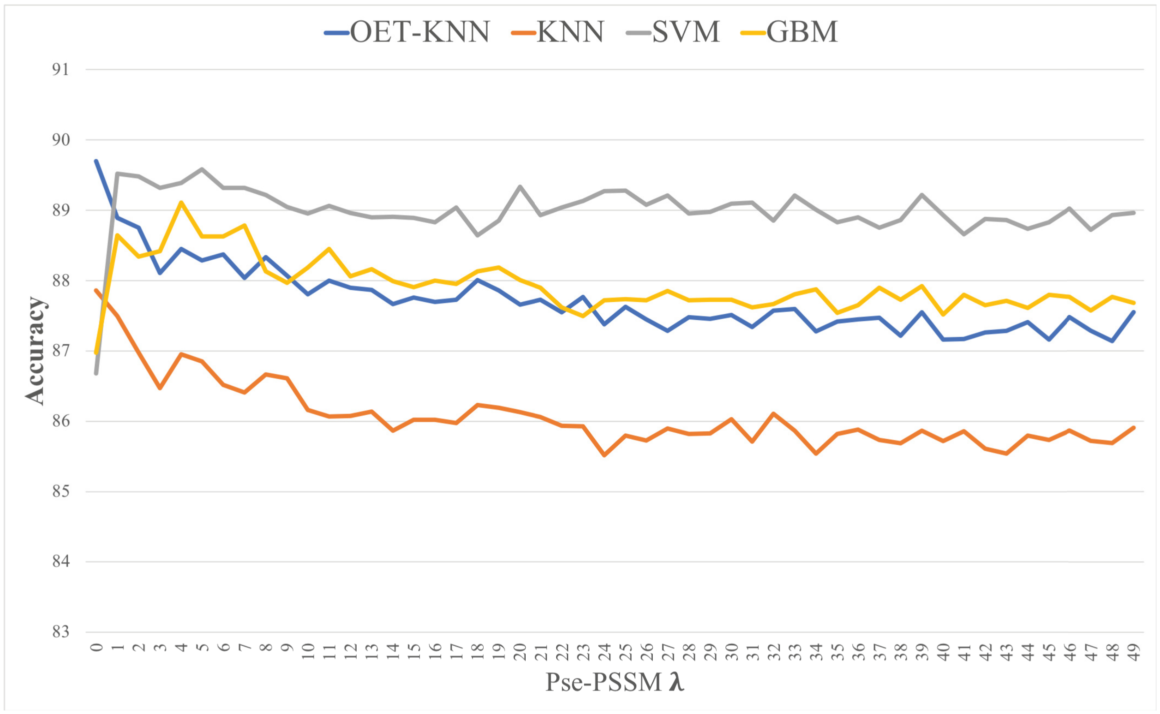


Figure 24: Choice of different models with Pse-PSSM, $\lambda \in (0, \ldots, 49)$

Table 36: LOOCV performances of the individual Pse-PSSM models

Encoding	**ML Algorithm**	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
Pse-PSSM, λ=0	OET-KNN	86.57	92.75	89.7	0.7953
	KNN	85.22	90.44	87.86	0.7580
	SVM	83.23	90.05	86.68	0.7350
	GBM	83.41	90.45	86.98	0.7409
Pse-PSSM, λ=1	OET-KNN	85.92	91.79	88.89	0.7788
	KNN	85.89	89.06	87.50	0.7501
	SVM	86.75	92.22	89.52	0.7912
	GBM	85.00	92.19	88.64	0.7744
Pse-PSSM, λ=2	OET-KNN	85.51	91.9	88.75	0.7762
	KNN	85.65	88.28	86.98	0.7397
	SVM	86.83	92.06	89.48	0.7904
	GBM	84.86	91.72	88.34	0.7682
Pse-PSSM, λ= 3	OET-KNN	84.69	91.44	88.11	0.7636
	KNN	84.97	87.92	86.47	0.7295
	SVM	86.97	91.61	89.32	0.7871
	GBM	85.01	91.74	88.42	0.7697
Pse-PSSM, λ=4	OET-KNN	85.46	91.37	88.45	0.7701
	KNN	85.44	88.41	86.95	0.7390
	SVM	86.87	91.85	89.39	0.7886
	GBM	85.71	92.41	89.11	0.7835
Pse-PSSM, λ=5	OET-KNN	85.17	91.32	88.29	0.7668
	KNN	85.60	88.07	86.85	0.7371
	SVM	86.82	92.26	89.58	0.7925
	GBM	85.10	92.08	88.63	0.7742
Pse-PSSM, λ=6	OET-KNN	85.26	91.39	88.37	0.7684
	KNN	85.41	87.61	86.52	0.7305
	SVM	86.72	91.85	89.32	0.7871
	GBM	84.96	92.22	88.63	0.7743
Pse-PSSM, λ=7	OET-KNN	84.95	91.06	88.04	0.762
	KNN	85.25	87.53	86.41	0.7281
	SVM	86.80	91.79	89.32	0.7872
	GBM	85.36	92.12	88.78	0.7771
Pse-PSSM, λ=8	OET-KNN	85.3	91.28	88.33	0.7676
	KNN	85.45	87.86	86.67	0.7335
	SVM	86.33	92.03	89.22	0.7853
	GBM	84.57	91.60	88.13	0.7641
Pse-PSSM, λ=9	OET-KNN	84.81	91.25	88.07	0.7626
	KNN	85.40	87.79	86.61	0.7322
	SVM	86.28	91.75	89.05	0.7819
	GBM	83.89	91.95	87.97	0.7614
Pse-PSSM, λ=10	OET-KNN	84.73	90.81	87.81	0.7572
	KNN	85.22	87.08	86.16	0.7232
	SVM	86.36	91.47	88.95	0.7796
	GBM	84.46	91.84	88.19	0.7655
Pse-PSSM, λ=11	OET-KNN	84.92	91.00	88.00	0.7611
	KNN	85.01	87.10	86.07	0.7214
	SVM	86.31	91.75	89.06	0.7821
	GBM	84.69	92.10	88.45	0.7707

LOOCV performances of the individual Pse-PSSM models (cont.)

Encoding	**ML Algorithm**	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
Pse-PSSM, λ=12	OET-KNN	84.59	91.11	87.90	0.7591
	KNN	85.00	87.13	86.08	0.7215
	SVM	86.26	91.59	88.96	0.7800
	GBM	84.24	91.77	88.06	0.7629
Pse-PSSM, λ=13	OET-KNN	84.80	90.87	87.87	0.7585
	KNN	84.67	87.57	86.14	0.7229
	SVM	86.42	91.32	88.90	0.7787
	GBM	84.62	91.60	88.16	0.7646
Pse-PSSM, λ=14	OET-KNN	84.54	90.72	87.67	0.7545
	KNN	84.92	86.78	85.87	0.7173
	SVM	86.49	91.27	88.91	0.7789
	GBM	84.33	91.55	87.99	0.7613
Pse-PSSM, λ=15	OET-KNN	84.64	90.79	87.76	0.7562
	KNN	84.56	87.45	86.02	0.7205
	SVM	86.22	91.49	88.89	0.7786
	GBM	84.32	91.42	87.91	0.7598
Pse-PSSM, λ=16	OET-KNN	84.66	90.66	87.70	0.7549
	KNN	85.15	86.87	86.02	0.7204
	SVM	86.03	91.55	88.83	0.7774
	GBM	84.15	91.75	88.00	0.7618
Pse-PSSM, λ=17	OET-KNN	84.82	90.56	87.73	0.7555
	KNN	84.97	86.96	85.98	0.7195
	SVM	86.56	91.46	89.04	0.7814
	GBM	84.51	91.30	87.95	0.7603
Pse-PSSM, λ=18	OET-KNN	84.64	91.30	88.01	0.7616
	KNN	84.72	87.70	86.23	0.7247
	SVM	86.17	91.05	88.64	0.7735
	GBM	84.48	91.68	88.13	0.7641
Pse-PSSM, λ=19	OET-KNN	84.96	90.70	87.86	0.7582
	KNN	85.45	86.91	86.19	0.7237
	SVM	86.07	91.55	88.85	0.7778
	GBM	84.68	91.60	88.19	0.7652
Pse-PSSM, λ=20	OET-KNN	84.46	90.78	87.66	0.7543
	KNN	84.76	87.47	86.13	0.7227
	SVM	86.36	92.23	89.33	0.7876
	GBM	84.17	91.76	88.01	0.762
Pse-PSSM, λ=21	OET-KNN	84.58	90.79	87.73	0.7556
	KNN	84.59	87.48	86.06	0.7212
	SVM	85.85	91.92	88.93	0.7796
	GBM	83.88	91.81	87.90	0.7598
Pse-PSSM, λ=22	OET-KNN	84.12	90.89	87.55	0.7523
	KNN	84.41	87.43	85.94	0.7189
	SVM	86.15	91.85	89.04	0.7817
	GBM	83.50	91.63	87.62	0.7543
Pse-PSSM, λ=23	OET-KNN	84.71	90.76	87.77	0.7565
	KNN	84.72	87.10	85.93	0.7186
	SVM	86.12	92.07	89.13	0.7837
	GBM	83.64	91.27	87.50	0.7518

LOOCV performances of the individual Pse-PSSM models (cont.)

Encoding	**ML Algorithm**	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
Pse-PSSM, λ=24	OET-KNN	84.61	90.08	87.38	0.7484
	KNN	84.33	86.69	85.52	0.7105
	SVM	86.34	92.13	89.27	0.7865
	GBM	83.69	91.65	87.72	0.7564
Pse-PSSM, λ=25	OET-KNN	84.15	91.01	87.63	0.7539
	KNN	84.51	87.07	85.80	0.7161
	SVM	86.44	92.04	89.28	0.7865
	GBM	83.79	91.59	87.74	0.7567
Pse-PSSM, λ=26	OET-KNN	84.34	90.47	87.45	0.75
	KNN	84.52	86.92	85.73	0.7147
	SVM	86.33	91.76	89.08	0.7825
	GBM	83.83	91.50	87.72	0.7561
Pse-PSSM, λ=27	OET-KNN	84.32	90.19	87.29	0.7468
	KNN	84.63	87.14	85.90	0.7181
	SVM	86.18	92.15	89.21	0.7852
	GBM	83.79	91.80	87.85	0.7589
Pse-PSSM, λ=28	OET-KNN	84.39	90.49	87.48	0.7506
	KNN	84.47	87.13	85.82	0.7164
	SVM	85.83	91.99	88.95	0.7802
	GBM	83.74	91.59	87.72	0.7562
Pse-PSSM, λ=29	OET-KNN	84.14	90.70	87.46	0.7504
	KNN	84.58	87.04	85.83	0.7166
	SVM	85.97	91.91	88.98	0.7806
	GBM	83.85	91.52	87.73	0.7565
Pse-PSSM, λ=30	OET-KNN	84.19	90.74	87.51	0.7514
	KNN	84.54	87.48	86.03	0.7208
	SVM	86.24	91.86	89.09	0.7827
	GBM	83.70	91.65	87.73	0.7565
Pse-PSSM, λ=31	OET-KNN	84.25	90.34	87.34	0.7477
	KNN	84.56	86.83	85.71	0.7142
	SVM	86.05	92.08	89.11	0.7832
	GBM	83.69	91.44	87.62	0.7541
Pse-PSSM, λ=32	OET-KNN	84.23	90.83	87.57	0.7527
	KNN	84.93	87.25	86.11	0.7222
	SVM	85.99	91.64	88.85	0.778
	GBM	83.75	91.48	87.67	0.7551
Pse-PSSM, λ=33	OET-KNN	84.28	90.84	87.60	0.7533
	KNN	84.32	87.39	85.87	0.7175
	SVM	86.18	92.15	89.21	0.7852
	GBM	84.02	91.50	87.81	0.7579
Pse-PSSM, λ=34	OET-KNN	84.02	90.46	87.28	0.7468
	KNN	84.25	86.78	85.54	0.7108
	SVM	86.17	91.77	89.01	0.7811
	GBM	83.69	91.96	87.88	0.7597
Pse-PSSM, λ=35	OET-KNN	84.30	90.45	87.42	0.7494
	KNN	84.67	86.93	85.82	0.7163
	SVM	85.75	91.84	88.83	0.7778
	GBM	83.74	91.23	87.54	0.7524

LOOCV performances of the individual Pse-PSSM models (cont.)

Encoding	ML Algorithm	Sensitivity	Specificity	Accuracy	MCC
Pse-PSSM, λ=36	OET-KNN	84.15	90.67	87.45	0.7503
	KNN	84.96	86.78	85.88	0.7176
	SVM	85.92	91.80	88.90	0.7789
	GBM	83.70	91.50	87.65	0.7549
Pse-PSSM, λ=37	OET-KNN	84.28	90.57	87.47	0.7504
	KNN	84.56	86.89	85.74	0.7148
	SVM	85.58	91.84	88.75	0.7761
	GBM	84.03	91.68	87.90	0.7598
Pse-PSSM, λ=38	OET-KNN	84.18	90.18	87.22	0.7453
	KNN	84.53	86.81	85.69	0.7137
	SVM	85.93	91.72	88.86	0.7783
	GBM	83.83	91.53	87.73	0.7563
Pse-PSSM, λ=39	OET-KNN	84.62	90.40	87.55	0.7519
	KNN	85.00	86.71	85.87	0.7173
	SVM	86.14	92.22	89.22	0.7855
	GBM	83.98	91.76	87.92	0.7603
Pse-PSSM, λ=40	OET-KNN	84.08	90.17	87.16	0.7443
	KNN	84.46	86.94	85.72	0.7144
	SVM	86.03	91.76	88.93	0.7796
	GBM	83.55	91.38	87.52	0.7522
Pse-PSSM, λ=41	OET-KNN	83.85	90.40	87.17	0.7446
	KNN	84.66	87.03	85.86	0.7172
	SVM	85.46	91.77	88.66	0.7744
	GBM	83.85	91.64	87.80	0.7578
Pse-PSSM, λ=42	OET-KNN	84.19	90.24	87.26	0.7461
	KNN	84.14	87.04	85.61	0.7123
	SVM	85.94	91.75	88.88	0.7787
	GBM	83.37	91.81	87.65	0.7551
Pse-PSSM, λ=43	OET-KNN	84.04	90.45	87.29	0.7469
	KNN	84.37	86.69	85.54	0.7109
	SVM	85.73	91.92	88.86	0.7784
	GBM	83.83	91.49	87.71	0.756
Pse-PSSM, λ=44	OET-KNN	84.24	90.50	87.41	0.7493
	KNN	84.78	86.80	85.80	0.7160
	SVM	85.88	91.53	88.74	0.7757
	GBM	83.56	91.55	87.61	0.7541
Pse-PSSM, λ=45	OET-KNN	84.02	90.22	87.16	0.7442
	KNN	84.56	86.89	85.74	0.7148
	SVM	85.80	91.77	88.83	0.7776
	GBM	83.83	91.66	87.80	0.7578
Pse-PSSM, λ=46	OET-KNN	84.32	90.56	87.48	0.7507
	KNN	84.51	87.20	85.87	0.7175
	SVM	85.97	91.99	89.02	0.7815
	GBM	83.73	91.70	87.77	0.7572
Pse-PSSM, λ=47	OET-KNN	84.03	90.47	87.29	0.747
	KNN	84.48	86.93	85.72	0.7145
	SVM	85.71	91.65	88.72	0.7755
	GBM	83.54	91.49	87.57	0.7532
Pse-PSSM, λ=48	OET-KNN	83.85	90.34	87.14	0.7439
	KNN	84.47	86.88	85.69	0.7138
	SVM	85.81	91.97	88.93	0.7798
	GBM	83.73	91.70	87.77	0.7572
Pse-PSSM, λ=49	OET-KNN	84.19	90.82	87.55	0.7522
	KNN	84.43	87.35	85.91	0.7183
	SVM	85.93	91.92	88.96	0.7803
	GBM	83.69	91.58	87.68	0.7556

Appendix B

Experiments on the psi-composition encodings for the prediction of other functional classes

Here, we conduct a preliminary study to determine how well the proposed psi-composition encodings can differentiate between the other functional classes and how this finding compares to that of other published work.

1 Dataset

The sequences in the *non-transporter* class of the transporter dataset in Chapter 5 were partitioned into other functional classes based on their biological roles: *enzyme*, *receptor*, or *other*. Protein sequences that have the annotation keyword "Receptor [KW-0675]" in the `Swiss-Prot` database were categorized into the *receptor* class, protein sequences that have an enzyme commission number (EC) annotation in the `Swiss-Prot` database were classified into the *enzyme* class, and all of the other sequences that were in neither the receptor class nor in enzyme class were labeled as *other*. The dataset was randomly partitioned (stratified by class) into a training set (90% of the data) and an independent testing set (10% of the data), as presented in Table 37

Class	**Training**	**Testing**	**Total**
Enzyme	2,591	287	2,878
Receptor	1,012	111	1,123
Other	2,340	263	2,603
Total	5,943	661	6,604

Table 37: Other dataset of functional membrane classes

2 Overview

We have a multiclass classification problem in which a membrane protein can be classified into a functional class (i.e., *receptor*, *enzyme*, or *other*). Similar to Chapter 5, an SVM with an RBF kernel was utilized as implemented by the `R` *e1071* library (version 1.6-8). The protein samples were represented using the psiPAAC encoding, as it achieved the highest performance during cross-validation. Since SVMs inherently deal with the binary classification problem, we extended the SVM approach using a one-against-the-rest approach, in which a binary classifier is trained for each individual class; samples of that class are positive samples and the rest of the samples are negative samples. The final prediction was determined by comparing the output probabilities of the positive class from each binary SVM, and the class with the highest probability was the predicted class. The best combination of the C and γ parameters was determined for each binary classifier independently by utilizing a grid search approach. Figure 25 delineates an overview of the prediction steps.

3 Comparison among different encodings

The overall accuracy and MCC of the SVM models for which a protein sample is encoded using different strategies is presented in Table 38. Similar patterns as in transporter detection (Chapter 5) were observed; as illustrated in Figure 26, the proposed psi-compositions outperformed the other variations, including the commonly applied PSSM

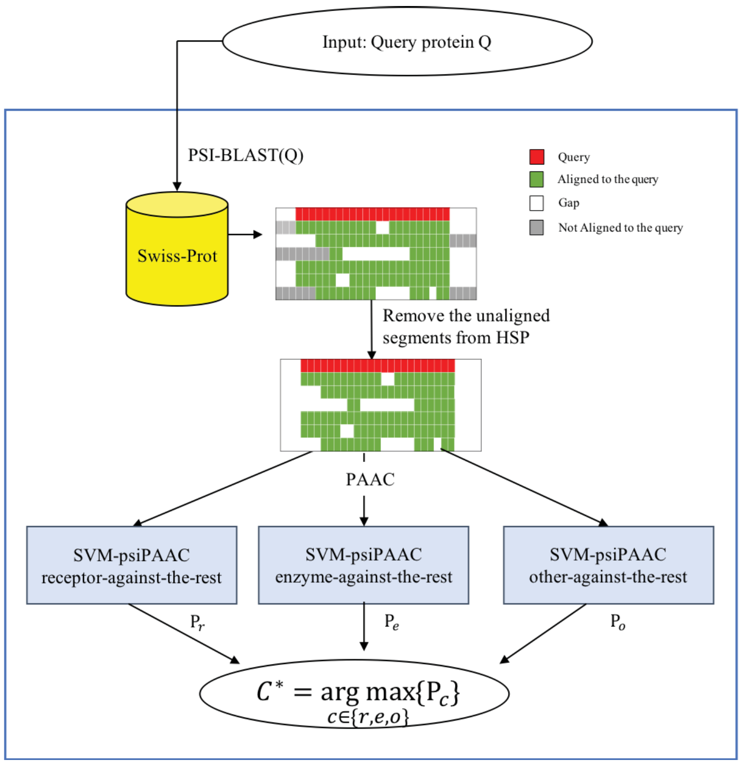

Input: Query protein Q
PSI-BLAST(Q)
Swiss-Prot
Query
Aligned to the query
Gap
Not Aligned to the query
Remove the unaligned segments from HSP
PAAC
SVM-psiPAAC receptor-against-the-rest
SVM-psiPAAC enzyme-against-the-rest
SVM-psiPAAC other-against-the-rest
P_r
P_e
P_o
$C^* = \arg\max_{c\in\{r,e,o\}}\{P_c\}$

Figure 25: TooT-REO overview
Given a query protein Q, the psiPAAC encoding is computed and input into three trained SVM classifiers, one for each functional type, namely, *receptors*, *enzymes*, and *others*. The output P_i indicates the probability that the query protein belongs to that class. The predicted class is the class with the highest output probability P_i.

encoding. The multiclass SVM that utilized psiPAAC encoding achieved the highest MCC in all of the classes; thus, this model was chosen as our functional membrane predictor. Table 39 shows the details of its cross-validation performance.

4 Comparison with other published work

To the best of our knowledge, there is no other tool that can simultaneously predict all of the functional membrane classes, so we cannot directly compare the performance of the proposed tool to that proposed by another work. Below, we perform two case studies — the first for detecting G protein-coupled receptors (GPCRs) and the second for enzyme detection.

Table 38: Functional class prediction of the different methods

Encoding	Accuracy	MCC
psiPAAC	**89.93 ± 0.14**	**0.8386 ± 0.0022**
blastPAAC	89.20 ± 0.20	0.8267 ± 0.0032
psiAAC	89.00 ± 0.06	0.8240 ± 0.0010
psiPseAAC	88.97 ± 0.12	0.8233 ± 0.0019
blastPseAAC	87.81 ± 0.18	0.8043 ± 0.0029
blastAAC	86.90 ± 0.17	0.7898 ± 0.0027
PSSM	86.09 ± 0.24	0.7767 ± 0.0039
PAAC	75.48 ± 0.25	0.6052 ± 0.0042
PseAAC	72.02 ± 0.14	0.5489 ± 0.0024
AAC	70.10 ± 0.26	0.5175 ± 0.0042

For each method, the table presents the accuracy and MCC as the $means \pm SDs$, calculated across the ten runs of the 10-CV in ascending order according to the MCC. The entry highlighted in bold refers to the best performance, that is, the model chosen for membrane functional prediction.

Table 39: Membrane functional prediction cross-validation performance

Class	Sensitivity	Specificity	Accuracy	MCC
Receptors	87.11 ± 0.32	98.67 ± 0.04	96.47 ± 0.05	0.8793 ± 0.0017
Enzymes	92.10 ± 0.13	93.56 ± 0.12	92.70 ± 0.11	0.8526 ± 0.0023
Others	88.75 ± 0.19	91.20 ± 0.16	90.21 ± 0.14	0.7961 ± 0.0029
Overall			89.93 ± 0.14	0.8386 ± 0.0022

Detailed cross-validation performance of the multiclass SVM where the protein samples were encoded using the psiPAAC encoding; this was the best-performing method during the cross-validation procedure.

4.1 Case 1: G protein-coupled receptors (GPCRs)

Studies on membrane receptor detection were designed to detect a specific function of the receptors, such as whether a membrane protein is a GPCR or a non-GPCR [XWC11] or an olfactory receptor or not [HS15]. Since GPCR proteins are frequent targets of therapeutic drugs, we compared the psiPAAC method to the state-of-the-art GPCR-2L predictor [XWC11] that achieved 97.2% accuracy in identifying proteins as GPCRs or non-GPCRs. Considering that our dataset contains a general receptor class and is not specific to GPCRs, we trained a new SVM to distinguish between GPCR and non-GPCR proteins using the psiPAAC encoding on the protein samples from the same dataset for the GPCR-2L predictor [XWC11]. Table 40 compares the performance. Overall, psiPAAC-SVM outperformed the GPCR-2L method by 0.04 (MCC).

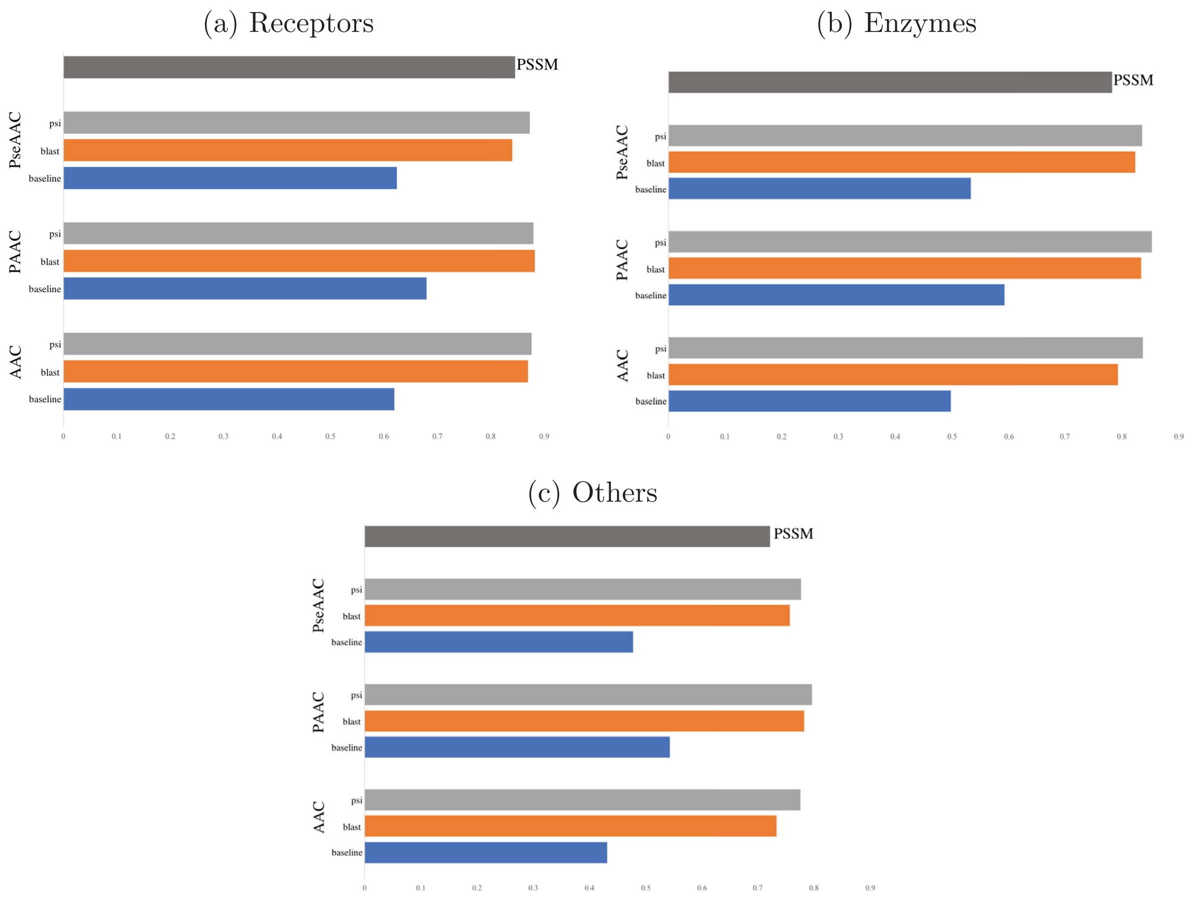


Figure 26: Impact of incorporating different sources of evolutionary information on membrane functional class prediction

The figure shows the improvement in the SVM performance, as measured by the MCC, of each functional class prediction when incorporating different sources of evolutionary information into the baseline encodings (AAC, PAAC, and PseAAC) and compares it to the commonly used encoding that also integrates evolutionary information (PSSM).

Table 40: Comparison with GPCR-2L in GPCR receptor detection

Class	**Number of proteins**	**correct prediction**		**Success rate (%)**		**MCC**	
		GPCR-2L	psiPAAC-SVM	GPCR-2L	psiPAAC-SVM	GPCR-2L	psiPAAC-SVM
GPCRs	367	360	359	98.09	97.82	0.93	0.97
Non-GPCRs	1,101	1,068	1,094	97.00	99.36	0.93	0.97
overall	1,468	1,428	1,453	97.28	98.97	0.93	0.97

This table compares the performance of the proposed method (psiPAAC-SVM) trained on the GPCR-2L dataset with the reported performance of the state-of-the-art GPCR-2L predictor [XWC11] trained on the same dataset.

4.2 Case 2: Enzymes

Among the membrane functional classes, enzymes are the most studied, mainly because enzymes are located in the membrane and there are many tools available to build powerful prediction models. There are several accessible web servers for enzyme protein detection in general; however, none of them are designed for membrane proteins. To see how our model compares to the state-of-the-art tools DEEPre [LWU+18] and EzyPred [SC07] in membrane enzyme detection, we ran our testing dataset on their web servers and compared the results to those achieved by our model. Table 41 compares the results and demonstrates that our psiPAAC-based membrane functional class predictor achieved the highest prediction accuracy and MCC among the examined enzyme predictors.

Table 41: Comparison with the state-of-the-art enzyme prediction tools

Class	**Number of proteins**	**correct prediction**			**Success rate (%)**			**MCC**		
		DEEPre	EzyPred	Proposed	DEEPre	EzyPred	Proposed	DEEPre	EzyPred	Proposed
Enzyme	287	239	233	255	83.28	81.18	88.85	0.75	0.21	0.84
Non-enzyme	374	342	155	355	91.44	41.44	94.92	0.75	0.21	0.84
overall	661	581	388	610	87.90	58.70	92.28	0.75	0.21	0.84

This table presents the performance of the proposed tool, psiPAAC-SVM, and compares it with the performance of the DEEPre [LWU+18] and EzyPred [SC07] web servers on the independent dataset from Table 10. The proposed tool achieved a higher success rate in both enzyme and non-enzyme detection than both DEEPre and EzyPred.

Appendix C

Detailed cross-validation performance in substrate specificity prediction

The following tables show the *mean*s ± *SD*s of the ten different runs of the 10-CV.

Substrate class	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
Nonselective	13.75 ± 3.95	99.59 ± 0.05	96.48 ± 0.20	0.2110 ± 0.0581
Water	37.92 ± 3.65	99.60 ± 0.06	97.25 ± 0.16	0.4748 ± 0.0372
Inorganic cations	87.08 ± 0.56	61.26 ± 0.88	64.61 ± 0.65	0.3364 ± 0.0124
Inorganic anions	13.80 ± 2.05	98.32 ± 0.31	87.68 ± 0.53	0.1716 ± 0.0349
Organic anions	32.06 ± 1.57	96.90 ± 0.22	87.18 ± 0.38	0.3060 ± 0.0176
Organo-oxygens	44.71 ± 2.91	92.42 ± 0.38	80.03 ± 0.61	0.3179 ± 0.0243
Amino acids and derivatives	30.99 ± 2.68	93.32 ± 0.50	79.93 ± 0.79	0.2109 ± 0.0281
Other organonitrogens	33.89 ± 1.79	95.56 ± 0.20	82.73 ± 0.36	0.3022 ± 0.0162
Nucleotides	41.36 ± 3.98	99.49 ± 0.10	97.32 ± 0.23	0.4724 ± 0.0397
Organic heterocyclics	23.82 ± 4.26	99.37 ± 0.11	95.43 ± 0.25	0.3194 ± 0.0440
Miscellaneous	10.91 ± 2.01	98.66 ± 0.24	87.22 ± 0.42	0.1539 ± 0.0370
Overall			52.21 ± 0.60	0.3628 ± 0.0091

Table 42: AAC performance

Substrate class	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
Nonselective	12.08 ± 5.36	99.74 ± 0.06	96.84 ± 0.24	0.2176 ± 0.0897
Water	32.92 ± 4.14	99.51 ± 0.08	97.08 ± 0.19	0.4094 ± 0.0402
Inorganic cations	88.37 ± 0.46	63.65 ± 0.61	67.30 ± 0.53	0.3915 ± 0.0114
Inorganic anions	8.15 ± 1.93	99.21 ± 0.19	88.84 ± 0.44	0.1481 ± 0.0420
Organic anions	38.76 ± 2.02	97.00 ± 0.38	88.52 ± 0.60	0.3758 ± 0.0233
Organo-oxygens	48.03 ± 2.41	92.70 ± 0.36	81.55 ± 0.81	0.3578 ± 0.0271
Amino acid and derivatives	32.25 ± 2.25	93.23 ± 0.54	80.75 ± 0.86	0.2259 ± 0.0270
Other organonitrogens	41.94 ± 1.58	96.07 ± 0.25	85.10 ± 0.37	0.3982 ± 0.0133
Nucleotides	42.73 ± 3.83	99.46 ± 0.06	97.43 ± 0.14	0.4774 ± 0.0313
Organic heterocyclics	22.35 ± 3.16	99.18 ± 0.18	95.27 ± 0.27	0.2804 ± 0.0317
Miscellaneous	21.62 ± 1.19	98.39 ± 0.17	88.48 ± 0.30	0.2797 ± 0.0149
Overall			54.80 ± 0.76	0.3999 ± 0.0108

Table 43: PseAAC performance

Substrate class	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
Nonselective	10.00 ± 3.51	99.71 ± 0.09	96.95 ± 0.16	0.1830 ± 0.0587
Water	93.33 ± 2.91	99.99 ± 0.03	99.78 ± 0.10	0.9607 ± 0.0174
Inorganic cations	88.98 ± 0.65	69.21 ± 0.96	71.91 ± 0.56	0.4745 ± 0.0096
Inorganic anions	24.24 ± 2.35	98.46 ± 0.26	90.06 ± 0.43	0.3130 ± 0.0316
Organic anions	38.76 ± 1.77	96.69 ± 0.40	88.86 ± 0.49	0.3666 ± 0.0193
Organo-oxygens	54.20 ± 1.51	93.83 ± 0.51	84.65 ± 0.67	0.4447 ± 0.0211
Amino acids and derivatives	47.39 ± 2.25	93.91 ± 0.26	84.40 ± 0.43	0.3815 ± 0.0193
Other organonitrogens	39.58 ± 2.68	95.54 ± 0.33	85.11 ± 0.36	0.3648 ± 0.0194
Nucleotides	68.64 ± 3.98	99.54 ± 0.10	98.41 ± 0.21	0.6901 ± 0.0371
Organic heterocyclics	27.35 ± 3.41	99.23 ± 0.08	95.85 ± 0.13	0.3390 ± 0.0297
Miscellaneous	11.01 ± 1.93	98.43 ± 0.20	88.24 ± 0.40	0.1475 ± 0.0358
Overall			58.93 ± 0.45	0.461 ± 0.0069

Table 44: PAAC performance

Substrate class	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
Nonselective	72.08 ± 4.41	99.73 ± 0.04	99.07 ± 0.10	0.7675 ± 0.0300
Water	95.83 ± 0.00	99.85 ± 0.00	99.73 ± 0.00	0.9376 ± 0.0000
Inorganic cations	93.77 ± 0.28	86.86 ± 0.49	88.45 ± 0.29	0.7748 ± 0.0053
Inorganic anions	59.24 ± 2.31	98.87 ± 0.12	95.48 ± 0.23	0.6610 ± 0.0200
Organic anions	68.04 ± 2.38	97.98 ± 0.11	95.08 ± 0.22	0.6727 ± 0.0175
Organo-oxygens	80.83 ± 0.36	98.61 ± 0.22	95.89 ± 0.23	0.8210 ± 0.0095
Amino acids and derivatives	80.07 ± 0.67	97.84 ± 0.17	95.23 ± 0.19	0.7782 ± 0.0081
Other organonitrogens	70.28 ± 2.14	95.81 ± 0.28	92.09 ± 0.26	0.6372 ± 0.0133
Nucleotides	81.82 ± 0.00	99.90 ± 0.07	99.52 ± 0.09	0.8719 ± 0.0219
Organic heterocyclics	72.35 ± 2.48	99.80 ± 0.13	98.91 ± 0.16	0.8032 ± 0.0277
Miscellaneous	46.57 ± 1.75	98.24 ± 0.17	93.58 ± 0.23	0.5269 ± 0.0186
Overall			79.84 ± 0.13	0.743 ± 0.0014

Table 45: TMC-AAC performance

Substrate class	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
Nonselective	63.33 ± 5.12	99.78 ± 0.06	98.93 ± 0.14	0.7220 ± 0.0411
Water	95.83 ± 0.00	99.93 ± 0.00	99.82 ± 0.00	0.9574 ± 0.0000
Inorganic cations	94.49 ± 0.59	84.49 ± 0.49	87.28 ± 0.36	0.7561 ± 0.0072
Inorganic anions	59.57 ± 1.76	99.43 ± 0.09	96.09 ± 0.16	0.7065 ± 0.0143
Organic anions	64.54 ± 1.55	98.94 ± 0.11	95.80 ± 0.14	0.7070 ± 0.0112
Organo-oxygens	80.19 ± 0.82	97.92 ± 0.23	95.09 ± 0.21	0.7887 ± 0.0081
Amino acids and derivatives	74.65 ± 1.29	98.55 ± 0.18	95.30 ± 0.23	0.7732 ± 0.0110
Other organonitrogens	72.50 ± 1.40	95.37 ± 0.48	91.88 ± 0.55	0.6390 ± 0.0207
Nucleotides	32.73 ± 4.18	99.79 ± 0.10	98.41 ± 0.12	0.4780 ± 0.0418
Organic heterocyclics	88.82 ± 4.11	99.87 ± 0.08	99.49 ± 0.15	0.9132 ± 0.0261
Miscellaneous	53.43 ± 1.30	98.18 ± 0.19	94.03 ± 0.25	0.5781 ± 0.0164
Overall			79.46 ± 0.30	0.7374 ± 0.0038

Table 46: TMC-PseAAC performance

Substrate class	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
Nonselective	75.00 ± 0.00	99.85 ± 0.05	99.30 ± 0.06	0.8185 ± 0.0141
Water	98.33 ± 2.15	99.85 ± 0.00	99.79 ± 0.05	0.9510 ± 0.0115
Inorganic cations	95.25 ± 0.28	88.26 ± 0.19	90.11 ± 0.14	0.8072 ± 0.0029
Inorganic anions	63.80 ± 1.99	98.80 ± 0.17	95.86 ± 0.23	0.6896 ± 0.0178
Organic anions	68.04 ± 1.75	97.65 ± 0.06	94.86 ± 0.18	0.6556 ± 0.0146
Organo-oxygens	83.63 ± 0.67	98.61 ± 0.18	96.35 ± 0.19	0.8397 ± 0.0079
Amino acids and derivatives	82.96 ± 1.28	98.49 ± 0.10	96.34 ± 0.16	0.8257 ± 0.0085
Other organonitrogens	66.39 ± 1.68	96.70 ± 0.27	92.67 ± 0.17	0.6412 ± 0.0084
Nucleotides	85.45 ± 1.92	99.96 ± 0.06	99.66 ± 0.07	0.9090 ± 0.0185
Organic heterocyclics	83.24 ± 4.81	100.00 ± 0.00	99.50 ± 0.14	0.9096 ± 0.0272
Miscellaneous	54.34 ± 1.33	98.09 ± 0.16	94.18 ± 0.16	0.5811 ± 0.0110
Overall			81.92 ± 0.12	0.7695 ± 0.0014

Table 47: TMC-PAAC performance

Substrate class	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
Nonselective	70.00 ± 2.64	99.66 ± 0.09	98.93 ± 0.13	0.7365 ± 0.0288
Water	100.00 ± 0.00	99.85 ± 0.00	99.82 ± 0.00	0.9599 ± 0.0000
Inorganic cations	92.88 ± 0.34	85.96 ± 0.70	87.52 ± 0.40	0.7562 ± 0.0071
Inorganic anions	54.57 ± 1.76	98.98 ± 0.23	95.21 ± 0.35	0.6346 ± 0.0278
Organic anions	64.43 ± 2.89	97.65 ± 0.17	94.42 ± 0.33	0.6294 ± 0.0245
Organo-oxygens	83.63 ± 0.31	98.06 ± 0.27	95.67 ± 0.26	0.8168 ± 0.0097
Amino acids and derivatives	82.68 ± 1.34	98.07 ± 0.18	95.75 ± 0.32	0.8049 ± 0.0149
Other organonitrogens	68.54 ± 0.98	96.35 ± 0.31	92.36 ± 0.29	0.6428 ± 0.0115
Nucleotide	70.45 ± 3.21	99.79 ± 0.11	99.16 ± 0.17	0.7698 ± 0.0415
Organic heterocyclics	70.00 ± 2.32	99.86 ± 0.04	98.90 ± 0.09	0.8000 ± 0.0178
Miscellaneous	49.39 ± 1.99	98.41 ± 0.17	93.94 ± 0.29	0.5602 ± 0.0229
Overall			79.33 ± 0.24	0.736 ± 0.0035

Table 48: TMC-TCS-AAC performance

Substrate class	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
Nonselective	58.33 ± 0.00	99.65 ± 0.04	98.67 ± 0.04	0.6545 ± 0.0089
Water	95.83 ± 0.00	99.87 ± 0.04	99.75 ± 0.04	0.9435 ± 0.0096
Inorganic cations	93.84 ± 0.29	85.50 ± 0.48	87.57 ± 0.33	0.7594 ± 0.0061
Inorganic anions	58.59 ± 1.49	98.90 ± 0.15	95.42 ± 0.13	0.6585 ± 0.0092
Organic anions	62.58 ± 1.46	97.46 ± 0.19	94.05 ± 0.29	0.6061 ± 0.0183
Organo-oxygen s	80.89 ± 0.79	98.67 ± 0.12	95.92 ± 0.11	0.8239 ± 0.0047
Amino acids and derivatives	78.24 ± 1.35	97.84 ± 0.12	94.98 ± 0.18	0.7660 ± 0.0095
Other organonitrogens	68.96 ± 1.39	97.15 ± 0.20	93.16 ± 0.22	0.6752 ± 0.0108
Nucleotides	76.82 ± 2.58	99.62 ± 0.06	99.06 ± 0.10	0.7618 ± 0.0232
Organic heterocyclics	75.00 ± 2.08	99.85 ± 0.05	99.04 ± 0.08	0.8294 ± 0.0145
Miscellaneous	48.89 ± 2.44	97.71 ± 0.18	93.16 ± 0.33	0.5158 ± 0.0255
Overall			79.03 ± 0.27	0.7324 ± 0.0037

Table 49: TMC-TCS-PseAAC performance

Substrate class	**Sensitivity**	**Specificity**	**Accuracy**	**MCC**
Nonselective	75.00 ± 0.00	99.78 ± 0.00	99.21 ± 0.00	0.7979 ± 0.0000
Water	95.83 ± 0.00	99.85 ± 0.00	99.74 ± 0.00	0.9376 ± 0.0000
Inorganic cations	95.19 ± 0.47	86.92 ± 0.28	89.36 ± 0.21	0.7936 ± 0.0046
Inorganic anions	64.35 ± 1.97	99.24 ± 0.18	96.38 ± 0.19	0.7252 ± 0.0155
Organic anions	68.04 ± 0.49	98.40 ± 0.13	95.66 ± 0.14	0.6974 ± 0.0084
Organo-oxygens	83.44 ± 0.52	98.97 ± 0.12	96.72 ± 0.15	0.8543 ± 0.0066
Amino acids and derivatives	84.08 ± 0.95	98.55 ± 0.16	96.56 ± 0.18	0.8357 ± 0.0085
Other organonitrogens	71.46 ± 0.95	96.84 ± 0.27	93.42 ± 0.22	0.6830 ± 0.0084
Nucleotides	80.91 ± 1.92	99.98 ± 0.04	99.61 ± 0.05	0.8904 ± 0.0132
Organic heterocyclics	82.35 ± 0.00	100.00 ± 0.00	99.47 ± 0.00	0.9050 ± 0.0000
Miscellaneous	55.96 ± 1.09	97.95 ± 0.16	94.21 ± 0.20	0.5858 ± 0.0136
Overall			82.53 ± 0.12	0.7772 ± 0.0019

Table 50: TMC-TCS-PAAC performance