A CLASSIFICATION SYSTEM FOR THE PROBLEM OF PROTEIN SUBCELLULAR LOCALIZATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

GÖKÇEN ALAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCES
IN
COMPUTER ENGINEERING

SEPTEMBER 2007

Approval of the thesis

"**A CLASSIFICATION SYSTEM FOR THE PROBLEM OF PROTEIN SUBCELLULAR LOCALIZATION**"

submitted by **Gökçen Alay** in partial fullfillment of the requirements for the degree of **Master of Sciences in Computer Engineering** by,

Prof. Dr. Canan Özgen _____
Dean, **Graduate School of Natural and Applied Sciences**

Prof. Dr. Volkan Atalay _____
Head of Department, **Computer Engineering**

Prof. Dr. Volkan Atalay _____
Supervisor, **Computer Engineering, METU**

Assist. Prof. Tolga Can _____
Co-supervisor, **Computer Engineering, METU**

Examining Committee Members:

Prof. Dr. Ismail Hakkı Toroslu _____
Computer Engineering, METU

Prof. Dr. Volkan Atalay _____
Computer Engineering, METU

Assist. Prof. Tolga Can _____
Computer Engineering, METU

Assist. Prof. Özlen Konu _____
Molecular Biology and Genetics, Bilkent University

Yıldıray Kabak (M.Sc.) _____
Computer Engineering, METU

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Gökçen Alay

Signature :

ABSTRACT

A CLASSIFICATION SYSTEM FOR THE PROBLEM OF PROTEIN SUBCELLULAR LOCALIZATION

Alay, Gökçen

M.S., Department of Computer Engineering

Supervisor: Prof. Dr. Volkan Atalay

Co-Supervisor: Assist. Prof. Tolga Can

September 2007, 85 pages

The focus of this study is on predicting the subcellular localization of a protein. Subcellular localization information is important for protein function annotation which is a fundamental problem in computational biology. For this problem, a classification system is built that has two main parts: a predictor that is based on a feature mapping technique to extract biologically meaningful information from protein sequences and a client/server architecture for searching and predicting subcellular localizations. In the first part of the thesis, we describe a feature mapping technique based on frequent patterns. In the feature mapping technique we describe, frequent patterns in a protein sequence dataset were identified using a search technique based on *a priori* property and the distribution of these patterns over a new sample is used as a feature vector for classification. The effect of a number of feature selection methods on the classification performance is investigated and the best one is applied. The method is assessed on the subcellular localization prediction problem with 4 compartments (Endoplasmic reticulum (ER) targeted, cytosolic, mitochondrial, and nuclear) and the dataset is the same used in P2SL. Our method improved the overall accuracy to 91.71% which was originally 81.96% by P2SL. In the second part of the thesis, a client/server architecture is designed and implemented based on Simple Object Access Protocol (SOAP) technology which provides a user-friendly interface for accessing the protein subcellular lo-

calization predictions. Client part is in fact a Cytoscape plug-in that is used for functional enrichment of biological networks. Instead of the individual use of subcellular localization information, this plug-in lets biologists to analyze a set of genes/proteins under system view.

Keywords: protein classification, subcellular localization, frequent pattern mining, cytoscape plug-in

ÖZ

PROTEINLERIN HUCRE ICI YERLESIMLERINI BULMAK ICIN BIR SINIFLANDIRMA SISTEMI

Alay, Gökçen

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Volkan Atalay

Ortak Tez Yöneticisi: Yar. Doç. Tolga Can

Eylül 2007, 85 sayfa

Bu çalışmanın odak noktası proteinlerin hücre içi yerleşimlerini bulmaktır. Hesaba dayalı biyolojide temel bir problem olan proteinlerin işlevlerinin belirlenmesinde, hücre içi yerleşim bilgisi önemlidir. Bu problem için, 2 ana bölümden oluşan bir sınıflandırma sistemi kuruldu: protein dizilerinden biyolojik olarak anlamlı verileri çıkarmak üzere tanımlanmış bir öznitelik eşleme yöntemine dayalı bir öngörücü ve hücre içi yerleşim bilgilerinin aranması ve öngörülmesi için inşa edilmiş bir istemci/sunucu mimarisi. Tezin ilk kısmında, yaygın örüntülere dayalı bir öznitelik eşleme yöntemi tanımlamaktayız. Tanımladığımız öznitelik eşleme yönteminde, yaygın örüntüler, bir protein veri kümesinden birincil özelliğe dayalı bir arama tekniği kullanılarak çıkartıldı ve bu örüntülerin yeni bir sekans üzerindeki dağılımları sınıflandırmada öznitelik vektörü olarak kullanıldı. Bir kaç öznitelik seçme metodunun sınıflandırma performansına etkisi araştırılarak, en iyi olan uygulandı. Metod, 4 bölümlü protein hücre içi yerleşimi öngörülmesi probleminde (Golgi aygıtına, stoplazmaya, mitokondriye ve çekirdeğe yerleşen) ve P2SL için kullanılan veri kümesi üzerinde değerlendirildi. Bizim metodumuz, yüzde 81.96 olan toplam doğruluk yüzdesini yüzde 91.71 e çıkardı. Tezin ikinci bölümünde, protein hücre içi yerleşim öngörülerine kullanıcı dostu bir arayüzle erişim sağlayan Basit Obje Erişim Protokolune (BOEP) dayalı bir istemci/sunucu mimarisi

tasarlandı ve gerçekleştirildi. Istemci tarafı aslında biyolojik ağlarin fonksiyonel zenginleştirilmesinde kullanılan bir Cytoscape eklentisidir. Bu eklenti, hücre içi yerleşim bilgisinin tek başına kullanılması yerine, biyologlara gen ya da protein kümelerini toplu biçimde analiz etme imkanı sunuyor.

Anahtar Kelimeler: protein sınıflandırması, hücre içi yerleşim, yaygın örüntü bulma, cytoscape eklentisi

ACKNOWLEDGMENTS

I would like to state my sincere gratitude to my supervisor Prof. Dr. Volkan ATALAY for his supervision, guidance and invaluable suggestions throughout the development of this thesis.
I deeply thank my co-supervisor Assist. Prof. Dr. Tolga CAN for his guidance and invaluable suggestions.
I deeply thank Assist. Prof. Dr. Rengül Çetin ATALAY for her "biological" guidance and suggestions.
I deeply thank Yıldıray Kabak (M.Sc.) for his assistance and patience.
I deeply thank the members of the Department of Computer Engineering and my friends for their suggestions during the period of writing the thesis.
This work is partially supported by TUBITAK EEEAG-105E035 and TUBITAK 2210-Scholarship program.
I deeply thank my family for their encouragement and their support.
And finally, it is my pleasure to express my deepest gratitude to my husband for his help, understanding and encouragement.

To my family

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES

LIST OF TABLES

TABLES

LIST OF SYMBOLS

S	Training data set
C	Label set corresponding to S
m	Cardinality of S and C sets
P'	Initial frequent pattern set
F'	Initial feature set
n	Cardinality of P' and F' sets
P	Final frequent pattern set
F	Initial feature set
t	Cardinality of P and F sets
v_i	Feature vector for sequence s_i
V	Feature vector set used in training
D	Set of qantification parameters

CHAPTER 1

INTRODUCTION

1.1 Problem Definition and Motivation

Eukaryotic cells are divided into several different compartments called subcellular localizations. After the translation process is completed, a protein is targeted to the appropriate compartment to perform the proper function. The operation environment of proteins is decided by the subcellular localizations. Thus, the subcellular localization of a protein is one of the key characteristics that determine its cellular function. Experimental analysis of subcellular localization is a costly and time consuming process. Therefore, automated tools for classifying proteins into their subcellular localizations are highly needed in order to annotate the newly discovered or unknown proteins. In the scope of this thesis work, a classification system is built that has two main parts: a subcellular localization prediction tool that is based on a feature mapping technique to extract biologically meaningful information from protein sequences and a client/server architecture for searching and predicting subcellular localizations.

1.1.1 Feature mapping technique

Classification of proteins into functional classes according to their primary sequences is an important problem in computational biology. Since, functionally important regions (catalytic sites, binding sites, structural motifs) are conserved over much wider taxonomic distances than the sequences themselves, conserved subsequences among different protein sequences are strong indicators of functional similarity [54].

Our assumption is that, in a set of protein sequences with the same subcellular localization, frequent subsequences, in fact a certain distribution of frequent subsequences determines the function or subcellular localization. The method described in this study uses the

distribution of frequent subsequences as features for classifying the given sequences into one of the subcellular localization classes. There are 4 kinds of subcellular localization classes: ER targeted (ER), mitochondrial (M), cytosolic (C) and nuclear (N).

In the first part of the thesis, a feature mapping technique is described. This technique is adapted from Birzele and Kramer [9] which was originally designed for protein secondary structure prediction. As a supervised learning method, discriminative features are first extracted from the training set. Using these features, test samples are mapped to the feature space and they are then classified. Mapping is performed based on frequent patterns of successive amino acids and there are three main steps. In the first step, frequent patterns in a protein sequence family are identified using level-wise string-mining algorithm [40]. At the second step, feature selection is performed by filtering where a statistical test is applied on the initial feature set. In fact, the effect of a number of feature selection methods on the classification performance is investigated and the best one is applied. Finally, a frequency based metric is used for feature quantification. Feature mapping of an input protein sequence is formed by the occurrence (frequency) of selected features. We assessed the method on the subcellular localization prediction problem with 4 compartments: ER targeted (ER), mitochondrial (M), cytosolic (C) and nuclear (N). The dataset is the same used in P2SL [4]. Our method improved the overall accuracy to 91.71% which was originally 81.96% by P2SL.

1.1.2 Client /server architecture

Subcellular localization prediction tools are in fact developed for biologists to ease their work. Therefore, providing a web based service is a must for the developers of prediction tools. Developing a user interface that is built on web browsers only let biologists access the prediction information and does not help on the usage or analysis of the data; leaves the rest of the work to the user. Most of the time, subcellular localization prediction information does not assist biologists as a single fact. Making analysis of a set of genes/proteins under a system view by integrating information from a number of domains provide more reliable information.

Cytoscape [47] is an open-source software program for network visualization and analysis. It provides basic functionality to visually integrate biomolecular interaction networks with databases of functional annotations and it is extensible through the plug-in architecture constructed under the core application. There are already a number of Cytoscape plug-ins for functional enrichment of biological networks with predicted or experimentally curated information [1].

In the second part of the thesis, a client/server architecture is designed and implemented based on Simple Object Access Protocol (SOAP) technology which provides a user-friendly interface for accessing the protein subcellular localization predictions. Client part is in fact a Cytoscape plug-in that is used for integrating functional annotations with biological networks. Instead of the individual use of subcellular localization information, this plug-in lets biologists to analyze a set of genes/proteins under system view.

1.2 An Overview of Related Systems and Contribution

Frequent patterns in biological sequences have been investigated for several reasons and they are often used for feature extraction in classification algorithms. Sometimes, focus of interest is not on the whole classification model but specifically on the pattern finding algorithm as it is in [20] and [34]. Ester and Zhang [20] introduce a top-down method for mining most specific frequent patterns. The use of concept graphs for extending the amino acid alphabet and the followed top-down strategy leads to find the most specific frequent patterns from which all other frequent patterns can be derived. Ye et al. [34] introduce novel approaches for efficient pattern mining. 3 types of patterns are defined in which certain kind of replacements and gaps are allowed and six different pattern mining algorithms are developed for their identification. For the evaluation of the discovered patterns, a number of reference patterns are computationally determined using multiple sequence alignments and they are used as gold standards.

Frequent patterns are used for numerous problems in bioinformatics. Birzele and Kramer [9] utilize frequent subsequences in the feature extraction stage of a secondary structure prediction algorithm. A level-wise string mining algorithm is used for extracting patterns. An extended alphabet definition is introduced which includes additional ten amino acid groups that share common chemical, structural and evolutionary properties. A frequency based metric, term-frequency inverted document frequency (TFIDF) is employed in the feature quantification stage. Zaiane et al. [46] use frequent subsequences for the protein localization prediction problem into two classes: intracellular and extracellular. Frequent patterns are extracted by using generalized suffix trees. Gaps are allowed in patterns but up to a defined maximum length. A binary representation is used in feature quantification. The described method which is a rule-based classification algorithm is compared with support vector machines (SVM) and Boosting methods. It is emphasized that results of a rule based classification algorithm is more understandable and readable compared to SVM where the

learned decision functions are often difficult for people to understand and extract further biological information other than the prediction result. She et al. [52] use frequent subsequences for the prediction of outer membrane proteins. The proposed association rule based method outperforms SVM in the experiments they performed. They also emphasize the advantage of a rule-based algorithm in providing biological insights that helps on the understanding of the structures and functions of the proteins.

Various methods have been introduced for predicting subcellular localization of proteins. TargetP [44] is one of the popular methods. It is a neural network-based prediction tool for subcellular location prediction. Using protein sequence information, it predicts over 4 localization classes: mitochondrion, chloroplast, secretory pathway, and 'other' localizations. For nonplant predictor predicts over 3 classes, excluding chloroplast localization. Besides the predicted localization class, TargetP outputs a reliability class for indicating the prediction quality. P2SL [4] is a system that predicts the subcellular localization of proteins in eukaryotic organisms based on implicit motif frequency distribution of protein sequences. P2SL is a hybrid computational system that predicts over ER targeted, cytosolic, mitochondrial and nuclear protein localization classes. Self-organizing maps (SOM) are used for feature extraction, and for classification a set of support vector machines (SVM) are employed. Only using protein sequence information, P2SL outputs one or more subcellular localization(s) as prediction with the computed possibilities associated to these localizations.

Based on a specific or a number of prediction methods, subcellular localization databases and web servers have been constructed that collects annotations of subcellular localization predictions. They can be large scale or specific to an organism. LOCtarget [42], SUBA [33] and DbSubLoc [60] are the recent ones. LOCtarget [42] is a web server and database that predicts and annotates subcellular localization for targets taken from TargetDB, a central registration database for structural genomics. It uses a combination of four different methods for prediction which use nuclear localization signals (PredictNLS), homology-based transfer of experimental annotations (LOChom), automatic text analysis of SWISS-PROT keywords (LOCkey) and neural networks (LOCnet). SUBA [33] is a database of predicted subcellular localizations for most Arabidopsis proteins. They combine various data sources including direct experimental datasets of MS, FP and AmiGO and Swiss-Prot and Description data for the annotation of proteins into 12 subcellular localizations. SUBA includes a web-browser based graphical user interface (GUI) like most of the prediction databases. DBSubLoc [60] differs from other databases on the user interface it provides. Instead of a web-browser based GUI, which is only useful for human accession, they develop a server/client suit, SubLoc [27],

that is based on SOAP technology to make easier the automated information retrieval done by machines. DbSubLoc SOAP server provides a number of functions including the prediction of subcellular localization for eukaryotic and prokaryotic proteins. They do prediction by support vector machines (SVM) [28] and PSORT [43].

Biological Networks Gene Ontology (BINGO) [58] and GOlorize [45] are two Cytoscape plug-ins entitled under the 'functional enrichment of networks' category. BINGO is a tool developed for determining significantly overrepresented Gene Ontology (GO) terms on subgraphs of biological networks visualized in Cytoscape. GOlorize is a plug-in for advanced network visualization, which use GO annotations with a class-directed layout algorithm to enhance visualization by exposing GO class structure on biological networks. Cerebral is a recently developed plug-in that works on interaction networks annotated with subcellular localization information. It is a plug-in for network visualization and it provides a layout that ease searching biological pathways or systems. Localization information does not need to be complete, since nodes without annotation can be positioned in the network by looking at the localizations of their neighbors.

In this thesis, frequent patterns are used for subcellular localization prediction problem of proteins into 4 compartments. Frequent pattern mining is done by a level-wise string mining algorithm [40]. The focus of the study is on the model described for classification, not on the pattern mining process. For the web end of the prediction tool, a client/server architecture is designed and implemented which is based on SOAP technology. Automated information retrieval is supported with the help of the functions provided by the SOAP server. In addition to that, a Cytoscape plug-in, *PlugP2SL*, is developed as a client application, which helps users to retrieve subcellular localization predictions and annotate biological networks visualized in Cytoscape with this information. PlugP2SL plug-in can be used in conjunction with Cerebral plug-in which provides a layout that ease searching biological pathways on interaction networks annotated with subcellular localizations. Predictions are performed by P2SL tool and MEP2SL database is used for retrieving stored predictions done by P2SL.

1.3 Organization of the Thesis

This thesis is organized in five main chapters, including this introduction chapter as the first chapter. Second, third and forth chapters are related with the first part of the thesis where a feature mapping technique is presented for the problem of protein subcellular localization prediction. In the second chapter, background information is given. In the third chapter

proposed method is presented. First, a general view of the system is given. Then main parts of the system are described in detail. The results obtained by the conducted experiments are presented in the forth chapter. Detailed explanation of the system performance on the dataset and comparisons with other prediction tools is given together with a discussion. In the fifth chapter design and implementation of the *PlugP2SL* plug-in is explained. First, background information is given. Next, P2SL plug-in design is explained in detail with use case, sequence and class diagrams. Finally, implementation details and usage of the plug-in are presented. The 6th and the last chapter contains the conclusion and the future work.

CHAPTER 2

BACKGROUND

2.1 Feature Selection

2.1.1 Basic concepts and definitions

Feature selection is one of the important issues in machine learning problems. Machine learning is an area of artificial intelligence and as the name implies, it allows computers to learn using example data or past experience to solve a given problem. In a typical supervised machine learning problem, the goal is to find a relationship between the data and the outcomes, given the training examples associated with the desired outcomes. In the process of achieving this goal, it is important to determine the features to use, and select the most discriminative, informative features among the possible ones.

Feature extraction problem can be decomposed into two steps: feature construction and feature selection [30]. For the aim of having a set of features representing the data points, at the feature construction step, one decides on the data representation, measurement techniques to use, and quantification schema to apply. If there are a few features in the feature set, and if there is no need to select among them, then feature extraction step terminates once the features are constructed. As the number of features increases, there will be a crucial need to apply a feature selection method. Although feature selection is primarily performed for selecting discriminative features, one can have other objectives to use a feature selection method, including:

1. visualization of the data, data understanding,
2. reduction of data collection time/cost, computation time; gaining efficiency,
3. improvement of prediction performance by defying the curse of dimensionality.

The *curse of dimensionality* is a term introduced by Bellman [7], and from the machine learning point of view, it is describing the problem caused by the exponential growth of the required number of samples with the increasing number of features to achieve the same predictive accuracy. Since in practice, number of training examples is fixed, using large number of features will degrade the classifier's performance.

In literature, both feature selection and feature extraction methods are referred as dimensionality reduction techniques. Given a set of features, feature selection problem is dealing with finding a subset of it. On the other hand, by feature extraction a set of measurements is mapped into a set of features. Taking a subset of a set can be viewed as a kind of transformation; therefore feature extraction definition subsumes feature selection definition. Feature selection is also referred as variable/attribute selection or variable/feature subset selection in the literature. Before going into a detailed description of feature selection methods, related definitions should be given.

Feature selection can be defined as a process that chooses a minimum subset of M features from the original set of N features($M \leq N$), so that the feature space is optimally reduced according to a certain criterion [25]. According to this definition the goal of feature selection is to find an optimal feature subset (maximizes the given criteria). Optimal feature subset is defined by Kohavi and John [51] as follows:

> **Definition 1 (Optimal feature subset)**
> *Given an inducer I, and a dataset D with features $X_1, X_2, \ldots X_n$ from a distribution D over the labeled instance space, an **optimal feature subset**, X_{opt}, is a subset of the features such that the accuracy of the induced classifier $C = I(D)$ is maximal.*

The criterion for the optimality is given as the classifier accuracy in this definition. Here, an optimal feature subset need not to be unique, since using different feature subsets, one could achieve the same accuracy. Finding X_{opt} is usually computationally intractable.

Relevancy and *non-redundancy* are two properties of a feature which are generally perceived as being necessary to obtain the optimal classification accuracy [15]. Their definitions and the arguments on the relation between them and optimal features are given in [51] and [37]. Kohavi & John showed by examples that:

- relevance does not imply optimality
- optimality does not imply relevance

Although there is no implication relation between relevance and optimality because of some exceptional examples (which are rare), these two concepts are related. In practice, by re-

moving the irrelevant features, suboptimal solutions can be reached for the optimal feature subset problem [30]. For the calculation of feature redundancy, feature correlation is widely used as a metric. If two features are completely correlated, then they are accepted to be redundant to each other. But, Guyon and Elisseeff [29] showed that correlation does not always imply redundancy.

2.1.2 Feature selection methods

There are basically three main categories for feature selection methods: filters, wrappers, and embedded methods [30].

Filters

Filters select important features as a pre-processing step before classification; therefore they are independent of the classifier being used. Variable ranking methods are the simplest form of the filter methods. By variable ranking methods, according to a scoring function, features are ranked, and the k highest ranked features are selected. Obtained feature subset is not usually optimal; however these methods are preferred because of their computational efficiency. Scoring functions used by variable ranking methods are explained in detail in Section 2.1.2.

Individual feature ranking methods have limitations, because they don't take into account feature dependencies [30]. In these methods, there is no need to know about other features while calculating a feature's score by which they are ranked. This kind of feature ranking methods called univariate methods. There are some multivariate methods which take into account feature dependencies. In practice, most of the feature sets contain dependencies. Therefore, assuming feature independence is a simplifying assumption and multivariate methods probably achieve better results since they don't make this assumption.

Two of the most well-known multivariate filter methods are RELIEF and FOCUS. Relief algorithm [35] uses a multivariate relevance criterion to rank individual features according to their relevance to the target concept. The algorithm is based on the *k-nearest-neighbor* approach. For each data point, the k closest point of the same class (nearest hits) and the k closest point of a different class (nearest misses) are selected. Each feature's score is computed by averaging the differences between the distances to the nearest hits and the distances to the nearest misses on the projection of the feature and taking their ratio [30]. A detailed algorithm is given at RELIEF Algorithm Section 3.2.3.

FOCUS algorithm [2] starts with an empty set of features and carries out an exhaustive search until it finds a minimal set of features. It is very sensitive to noise and because of the conducted exhaustive search, it is impractical to use for domains containing more than 25-30 features [15].

Wrappers

Wrappers search the space of feature subsets by using the prediction performance of the given learning machine as a scoring function until there is no feature subset which has a score better than the one achieved [51]. According to their search strategies, they can be divided into two classes: backward elimination and forward elimination. One can obtain better results with wrapper methods than filter methods. However, wrapper methods are very expensive in terms of computational cost. Wrapper methods are also criticized for producing feature subsets only specific to a given learning machine, not the optimal feature subset which is independent of the classifier used [29].

Embedded methods

Embedded methods perform feature selection in the training phase. In other words learning process implicitly comprises feature selection. This is the point where embedded methods differ from the other two feature selection methods. Filter methods are not involved in the learning process, their role finishes before the learning process begins. Wrapper methods use a learning machine, actually its prediction performance for assessing the quality of the feature subsets, but feature selection is not involved in the learning process, it is performed after feature selection step finishes. In contrast to filter and wrapper methods, embedded methods incorporates learning and feature selection processes. Decision trees, boosting and an extension of SVMs which is called *recursive feature elimination* - SVM (RFE-SVM) can be given as examples of embedded methods.

2.2 Association Measures

Association measures are mathematical formulae that compute an association score which indicates the amount of (statistical) association between the subjected variables [22]. An association measure can be one-sided or two-sided depending on whether it distinguishes between positive and negative association or not.

For one-sided association measures, high scores indicate strong positive association and low scores mean that either there is negative association or the components are independent. On the other hand, high scores indicate any kind of strong association (positive or negative) for two-sided association measures, and low scores mean that the components are independent regardless of the sign of the score.

Association scores are computed by different measures such as measures based on statistical hypothesis tests, maximum-likelihood estimates for various coefficients of association strength, measures from information theory and measures based on heuristic combinations of the observed joint and marginal frequencies [22]. In general, the association scores computed by different measures cannot be compared directly. Since their typical usage is for ranking pair types, direct comparison of the scores is not necessary most of the time.

Association measures are generally defined on datasets that are represented by contingency tables. Before going into details, it is worth to introduce contingency tables.

2.2.1 Contingency tables

Contingency tables are statistical tables that show the frequencies of data elements in which the row entries tabulate the data according to one variable and the column entries tabulate it according to another variable [59]. Contingency tables are used to record and analyse the relationship between two variables. In the sequel, we follow the notation of von Evert [22]. In Figure Table 2.1, data is classified by two variables: U and V. For simplification, they

Table 2.1: (a) Contingency table of observed frequencies (b) Contingency table with marginal frequencies.

	$V = v$	$V \neq v$
$U = u$	O_{11}	O_{12}
$U \neq U$	O_{21}	O_{22}

$O_{11} + O_{12} + O_{21} + O_{22} = N$

(a)

	$V = v$		$V \neq v$	
$U = u$	O_{11}	$+$	O_{12}	$= R_1$
	$+$		$+$	
$U \neq U$	O_{21}	$+$	O_{22}	$= R_2$
	$= C_1$		$= C_1$	

(b)

are chosen to be categorical variables with two categories. The cell counts $O_{11}, \ldots, O_{22}$ of the contingency table are called the observed frequencies and they add up to the sample size

N. The row sums R_1, R_2 and column sums C_1, C_2 are referred to as *marginal frequencies*. O_{11} is also called the joint frequency of the pair type (u, v).

2.2.2 Random sample model

Statistical interpretation of data summarized by a contingency table is based on a random sample model, which assumes that the observed data is a sample and it is drawn randomly from an infinite population. Such an infinite population can be described by a model with parameters. The sample, which is the real data at hand, can then be examined in terms of its convenience to the assumed model. For the case of co-occurrence data, we need a model described with a set of random variables and their relative probability distribution in the population. For being consistent with Section 2.2.1, the dataset described with two variables U and V will be used. Random sample model contains two random variables, U and V and the sampling distribution is determined by the *probability parameters* $\tau_{11}, \ldots, \tau_{22}$ which are described in the following equations.

$$\begin{aligned} P(U = u \wedge V = v) &= \tau_{11} \\ P(U \neq u \wedge V = v) &= \tau_{21} \\ P(U = u \wedge V \neq v) &= \tau_{12} \\ P(U \neq u \wedge V \neq v) &= \tau_{22} \end{aligned}$$

Since probability parameters add up to one, only three of the four probability parameters are free. Therefore, it is more convenient to use an equivalent set of three parameters, given by the equations:

$$\begin{aligned} \pi &= P(U = u \wedge V = v) = \tau_{11} \\ \pi_1 &= P(U = u) = \tau_{11} + \tau_{12} \\ \pi_2 &= P(V = v) = \tau_{11} + \tau_{21} \end{aligned}$$

Although it is obvious that the amount of association between the variables U and V depends one these three probability parameters, there is no exact way of combination of the probability parameters to form an association strength coefficient. It is clear that, having a larger value of π and smaller values of π_1 and π_2 indicates stronger association. Two mostly used combinations are as follows:

- *mu-value* coefficient, $\mu = \frac{\pi}{\pi_1 \pi_2}$
- *odds ratio* coefficient, $\theta = \frac{\tau_{11}\tau_{22}}{\tau_{12}\tau_{21}}$

Maximum-likelihood estimates for this kind of coefficients of association strength are used as an association measure [21]. These measures will be investigated in Section 2.2.8.

2.2.3 Statistical inference by maximum-likelihood estimates of population parameters and hypothesis tests

Using observed data, one can make inferences about the population parameters. The simplest form of inference is direct "point" estimates for the probability parameters, which maximize the likelihood of the observed contingency table among the set of all parameter values. These kinds of estimates are known as *maximum-likelihood estimates* (MLE). MLE for probability parameters π, π_1 and π_2 are given by frequencies p, p_1 and p_2 in the following equations:

$$\begin{aligned} p &= \frac{O_{11}}{N} \\ p_1 &= \frac{O_{11}+O_{12}}{N} = \frac{R_1}{N} \\ p_1 &= \frac{O_{11}+O_{21}}{N} = \frac{C_1}{N} \end{aligned}$$

Another approach to statistical inference compares observed relative frequencies with the expected relative frequencies by assuming a model, given some hypothesis about the parameter values of the model. If the observed data correspond to an unlikely outcome of the assumed model, the null hypothesis is rejected [21]. This procedure is called a *statistical hypothesis test.*

2.2.4 Null hypothesis of independence

Most of the hypothesis tests that are used for investigating cooccurence data are based on the *null hypothesis of independence*, H_0, whose definition is derived from a well-defined concept for the complete absence of association: statistical independence, as seen below:

$$H_0 : \pi = \pi_1 \cdot \pi_2 \approx p_1 \cdot p_2$$

Since the hypothesis contains parameter values whose values are not exactly known, maximum-likelihood estimates of these parameters are used in the equation, and so point hypothesis is formed. Using this hypothesis, expected frequency values are calculated as it is given in Figure Table 2.2 and they are compared with the observed frequencies to reject or accept the hypothesis. In other words the amount of evidence against the null hypothesis of independence is used as an association score. This amount can be quantified by the likelihood of the observed data or by the $p - value$ of a statistical hypothesis tests [21]. Since both of these

values are probability values in the range $[0, 1]$, smaller values indicate strong association. If the negative base 10 logarithm of the likelihood or $p - value$ is used as an association score, then larger values indicate strong association.

Table 2.2: Expected frequencies.

	$V = v$	$V \neq v$
$U = u$	$E_{11} = \frac{R_1 C_1}{N}$	$E_{12} = \frac{R_1 C_2}{N}$
$U \neq U$	$E_{21} = \frac{R_2 C_1}{N}$	$E_{22} = \frac{R_2 C_2}{N}$

2.2.5 Likelihood measures

Likelihood measures use the probability of the observed frequencies as the amount of evidence against the null hypothesis of independence. They are two-sided measures, and some of them are given in Equations (2.1), (2.2).

$$\textbf{Poisson likelihood} = e^{E_{11}} \frac{(E_{11})^{O_{11}}}{O_{11}!} \tag{2.1}$$

$$\textbf{Poisson stirling}_{\textbf{log}} = O_{11} \cdot (\log O_{11} - \log E_{11} - 1) \tag{2.2}$$

Poisson-Stirling measure (Equation 2.2) is an approximation to the negative logarithm of Poisson-likelihood measure. Poisson-Stirling measure is described by Quasthoff and Wolff [49] and it is the most widely used one among the other likelihood measures.

2.2.6 Exact hypothesis tests

Since likelihood measures consider only a single outcome, which is the observed contingency table, their values cannot be used as a direct measure of the amount of the evidence against H_0. But, on the other hand, statistical hypothesis tests compute a measure called p-value which is the total probability of all possible outcomes that are similar to or more "extreme" than the observed contingency table [21]. Low values of p-value indicate significant evidence against the null hypothesis, and so its negative base 10 logarithm can be used as an association score.

Among exact hypothesis tests Fisher's exact test (Equeation 2.3) is accepted as the most appropriate test for independence in 2-by-2 contingency tables[68]. But, for large samples, none of the exact hypothesis tests is computationally intractable.

$$\textbf{Fisher} = \sum_{k=O_{11}}^{min\{R_1,C_1\}} \frac{\binom{C_1}{k} \cdot \binom{C_2}{R_1-k}}{\binom{N}{R_1}} \tag{2.3}$$

2.2.7 Asymptotic hypothesis tests

Instead of the exact $p-value$, asymptotic hypothesis tests use a much simpler association score called test statistic. The distribution of the test statistic under the null hypothesis converges to a known limiting distribution for $N \to \infty$, and the limiting distribution can then be used to approximate the $p-value$.

z-score and t-score

The $z-score$ measure (Equation 2.4) is an asymptotic version of the binomial exact hypothesis test. It was used by Dennis [17] and later by Berry-Rogghe [8]

$$\mathbf{z-score} = \frac{O_{11} - E_{11}}{\sqrt{E_{11}}} \tag{2.4}$$

When the expected frequency E_{11} is small, $z-score$ values smaller expected frequency pairs become can become very large. By estimating the variance from the sample rather than using the expected value, Church et al [11] obtain better results from $Student'st-test$. The resulting test statistic is called $t-score$. The equation is given below.

$$\mathbf{t-score} = \frac{O_{11} - E_{11}}{\sqrt{O_{11}}} \tag{2.5}$$

Both $z-score$ and $t-score$ are one-sided measures.

Pearson's chi-squared test

In the field of mathematical statistics, the standard asymptotic hypothesis test for independence in 2-by-2 contingency tables is Pearson's chi-squared test [16]. Its test statistic is a kind of mean square error between the observed frequencies O_{ij} and the expected frequencies E_{ij} scaled by the expected variances of the cell frequencies. In the limiting case, the test statistic of Pearson's chi-squared test has an asymptotic χ^2 (chi-squared) distribution with one degree of freedom. The test is two-sided with non-negative association scores. The

equation is given in two forms; the first one is the one which Edmundson [19] suggested in 1965 and the former/latter one is its normal form.

$$\mathbf{chi-squared_i} = \sum_{i,j} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{2.6}$$

$$\mathbf{chi-squared} = \frac{N(O_{11} - E_{11})^2}{E_{11}E_{22}} \tag{2.7}$$

Likelihood ratio test

Another class of test statistics is likelihood ratio tests, which are based on the ratio between the maximum probability of the observed data under two different hypothesis; null hypothesis, no hypothesis (null hypothesis constraint relaxed). The use of $log-likelihood$ as an association measure was originally suggested by Dunning [18]. Here is the standard form of the $log-likelihood$ association measure:

$$\mathbf{log-likelihood} = 2\sum_{ij} O_{ij} \log \frac{O_{ij}}{E_{ij}} \tag{2.8}$$

2.2.8 Point estimates of association strength

As it is introduced in Section 2.2.2, the two most common coefficients of association strength are the mu-value, μ and the odds ratio, θ. Church and Hanks [10] used the $maximum-likelihood$ estimate for the logarithm of μ as an association measure, which is mutual information (MI), a concept from information theory.

$$\mathbf{MI} = \log \frac{O_{11}}{E_{11}} \tag{2.9}$$

Maximum likelihood estimate of θ which is called the odds-ratio measure is as follows:

$$\mathbf{odds-ratio} = \log \frac{O_{11}O_{22}}{O_{12}O_{21}} \tag{2.10}$$

The value of odds-ratio is undefined when O_{12} or O_{21} is zero. A discounting technique can be used to avoid such scores, by adding 0.5 to each observed frequency before the calculation of the ratio [21]

$$\mathbf{odds-ratio_{disc}} = \log \frac{(O_{11} + 1/2)(O_{22} + 1/2)}{(O_{12} + 1/2)(O_{21} + 1/2)} \tag{2.11}$$

Liddell [39] suggests a different association strength coefficient where the difference of the column proportions is used. Liddell coefficient and its maximum likelihood estimate are given below:

$$\mathbf{Liddell} = \frac{N(O_{11} - E_{11})}{C_1C_2} = \frac{O_{11}O_{22} - O_{12}O_{21}}{C_1C_2} \tag{2.12}$$

It is clear that two parameter ratios have to be combined in some way to obtain an association strength coefficient. Different link functions are used for this combination; such as minimum is used as a link function to obtain minimum sensitivity (MS) measure [48], geometric mean is used to obtain gmean measure, harmonic mean is used to obtain Dice coefficient [56]. The best-known coefficient from this group is the Dice coefficient, which is as follows:

$$\textbf{Dice} = \frac{2O_{11}}{R_1 + C_1} \tag{2.13}$$

2.2.9 Heuristic measures

There are some association measures rely on heuristics or heuristic variants of other measures. The simplest possible association measure that can be categorized in this section is the frequency of the pair types.

$$\textbf{Frequency} = O_{11} \tag{2.14}$$

MI^2 and MI^3 are heuristic variants of MI. They both try to increase the influence of O_{11} in the numerator to reduce the overestimation of low-frequency data. MI^3 measure was suggested by Daille [13], because of the fact that it is the one that gives the best performance among the possible versions of MI with $(O_{11})^k$ in the numerator for $k = 2 \ldots 10$.

$$\textbf{MI}^{\textbf{2}} = \log \frac{(O_{11})^2}{E_{11}} \tag{2.15}$$

$$\textbf{MI}^{\textbf{3}} = \log \frac{(O_{11})^3}{E_{11}} \tag{2.16}$$

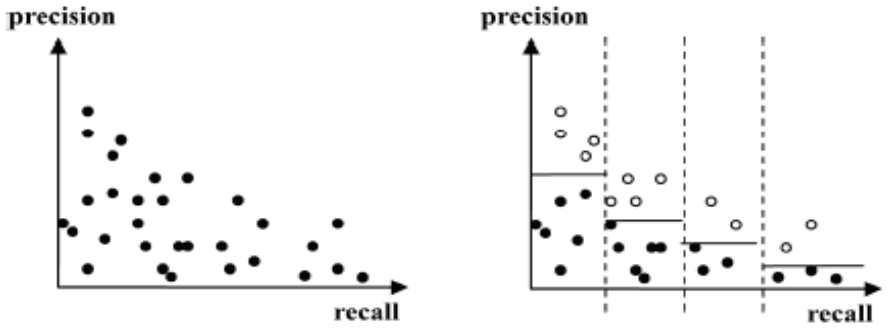
2.2.10 Precision-recall binning

Precision and recall are common measures of interestingness of a feature and can be defined from Table Table 2.1 as follows.

$$\textbf{precision} = \frac{O_{1i}}{R_1} \tag{2.17}$$

$$\textbf{recall} = \frac{O_{1i}}{C_i} \tag{2.18}$$

Precision-recall binning method's basic principle is shown in Figure Figure 2.1. Given the precision and recall values of all features with respect to a given class, the recall dimension is subdivided into equal width intervals (bins) and in terms of precision, the top percent of features is selected from each bin [9]. For binary classification problems, the union of the sets which are determined to be significant for each class can be taken as the final feature set.



(a) Precision recall values of all feature.

(b) Features above the solid lines are selected.

Figure 2.1: Precision-recall binning method (adopted from [9]).

2.3 Boosting Approach and the Adaboost Algorithm

Boosting is a machine learning approach for performing supervised learning. It is a general method of combining rough rules of thumb for obtaining a highly accurate prediction rule. Boosting occurs in iterations; at each iteration a weak learner learns a distribution of training data (data is weighted with a distribution). After determining the weight of the weak learner by considering its learning accuracy, the weak learner is added to the final strong learner. At the beginning of each iteration, training data is reweighed in such a way that misclassified data points gain weight and correctly classified ones lose weight in order to make the next weak learner focus on the most misclassified data points by the previous weak learners.

There are many boosting algorithms. They mainly vary in the method they use for weighting training data points and weak learners. AdaBoost algorithm which is introduced in 1995 by Freund and Schapire [23] is the most popular one among the other boosting algorithms. Pseudocode for AdaBoost algorithm is given in Figure Figure 2.2.

In the given AdaBoost algorithm, iteration starts at the time point 1 (when $t = 1$). The vector $D_t(i)$ which contains weights of training examples is initialized with equal weights. At each iteration weak learner finds a weak hypothesis h_t with error e_t. The goodness of a weak hypothesis is measured by its error and quantified by the given formula for α_t. Weight vector for training examples, $D_t(i)$, is updated in a way that the weights of the incorrectly classified examples are increased, and the weights of the correctly classified examples are decreased. At the end of T iterations, weighted sum of the entire weak hypothesis is taken to form the final hypothesis.

Given: $(x_1, y_1), \ldots, (x_m, y_m)$
where $x_i \in X,\ y_i \in Y = \{-1, 1\}$
Initialize $D_1(i) = 1/m$
For $t = 1 \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1,\ +1\}$ with error

$$\epsilon_t = Pr_{i \sim D_t} \left[h_t(x_i) \neq y_i \right]$$

- Choose $\alpha_t = \frac{1}{2} \ln \left(\dfrac{1 - \epsilon_t}{\epsilon_t} \right)$
- Update:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} \epsilon^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ \epsilon^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \end{aligned}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = sign\Big(\sum_{t=1}^{T} \alpha_t h_t(x) \Big)$$

Figure 2.2: The boosting algorithm, AdaBoost (adopted from [24]).

As mentioned in Section 2.1.2, AdaBoost algorithm is categorized under the embedded feature selection methods. If the weak learner is chosen as a tresholding method on a selected feature, at each iteration the best feature and the threshold value is chosen as the weak hypothesis. Being a combination of weak hypothesis, the final hypothesis can then be seen as the selected feature subset. With a constraint on the weak learner, AdaBoost algorithm can be seen as a learning algorithm in which an implicit feature selection process is embedded.

2.4 Support Vector Machines

Support Vector Machines are a set of supervised learning algorithms that can perform classification and regression. They belong to a family of generalized linear classifiers and they perform the structural risk minimization principle. In a linearly separable dataset, there is usually more than one linear classifier which correctly classifies the set. Following the results of statistical learning theory, by using SVM, one can minimize the empirical risk and maximize the geometric margin simultaneously. Therefore among all possible separating hyperplanes, the one with the largest margin, in other words the hyperplane that maximizes the distance between itself and the nearest vectors from each class is chosen. This hyperplane is called the optimal hyperplane. In the sequel, we follow the notation of von Vert [63].

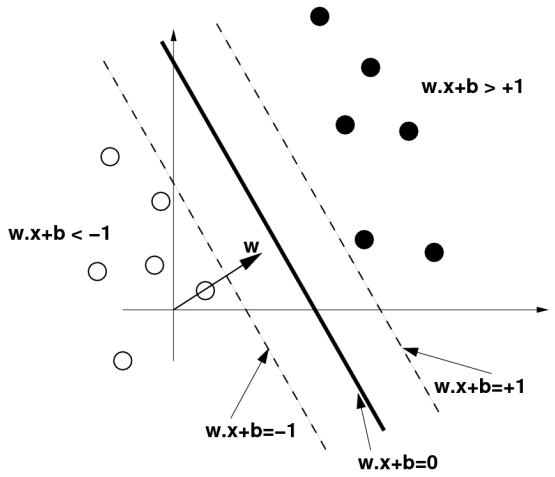


Figure 2.3: A separating hyperplane (adopted from [63]).

A separating hyperplane is given in Figure Figure 2.3. Consider there is a training data

set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ where x_i is a data point and y_i is the corresponding class label where $y_i \in \{+1, -1\}$. The pair $(\vec{w}, b)$ must satisfy the following equation for the given separating hyperplane being an optimal hyperplane:

$$\begin{cases} \vec{w} \cdot \vec{x_i} + b \geq 1 & \texttt{if } y_i = +1 \\ \vec{w} \cdot \vec{x_i} + b \leq -1 & \texttt{if } y_i = -1 \end{cases} \tag{2.19}$$

and for which the norm $\|\vec{w}\|$ is minimum. This is actually a constrained optimization problem. For the solution of constrained optimization problems, introducing the Lagrangian function and using the dual form of the equation are classical approaches. For the details of the solution of this constrained optimization problems, please refer to [63]. Here is the decision function of SVM where the mapping is done via a kernel function $K(x_i, x_j)$:

$$f(x) = sgn(\sum_{i=1}^{N} y_i \alpha_i \cdot K(x, x_i) + b) \tag{2.20}$$

where α_i can be obtained solving the following equation:

$$\text{Maximize } \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \cdot y_i y_j \cdot K(x_i, x_j) \tag{2.21}$$

$$\text{subject to } 0 \leq \alpha_i \leq C, \sum_{i=1}^{N} \alpha_i c_i$$

In the equation 2.21, C controls the trade-off between training error and margin. Support vectors are the closest vectors to the optimal hyperplane from each class and they can be defined using equation 2.20. The x_i with corresponding $\alpha_i > 0$ are called the support vectors.

There are several different types of kernel functions. These include linear, polynomial, radial basis function (RBF) and sigmoid. The RBF is one of the most popular kernels used in SVM. Its equation is given in Equation 2.22.

$$K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2} \tag{2.22}$$

CHAPTER 3

SYSTEM AND MODULES

3.1 System

Classification of proteins using only primary sequences is often performed by transferring annotations after sequence alignment. Similarity between proteins can also be identified by examining conserved regions. Conserved regions (patterns or motifs) are shared subsequences among different protein sequences and they are strong indicators of functional similarity. Our assumption is that, in a set of protein sequences targeted to the same subcellular localization, a certain distribution of frequent subsequences determines the function or subcellular localization. The method described in this study uses the distribution of frequent subsequences as features for classifying the given sequences into one of the subcellular localization classes. There are 4 kinds of subcellular localization classes: ER targeted (ER), mitochondrial (M), cytosolic (C) and nuclear (N).

In this part of the thesis, we describe a feature mapping technique based on frequent subsequences to extract biologically meaningful information from protein sequences for classification purposes. The feature mapping technique is adapted from Birzele and Kramer [9] which was originally designed for protein secondary structure prediction. As a supervised learning method, discriminative features are first extracted from the training set. Using these features, protein sequences are mapped to the feature space and they are then classified by the classifiers which are trained with the training examples.

As it is observed in Figure Figure 3.1, the system for the prediction of subcellular localization is composed of classification modules and a decision combination module. Classification modules are forming the core of the system. Since each module contains a binary classifier, in order to cover different combinations of pairs of 4 classes, 6 classification modules are used in the system. These modules are responsible for investigating the given protein sequence

and labeling it with one of the two classes. Sub-decisions given by the classification modules are combined by Decision Combination Module and the final decision is formed. Decision Combination Module is basically performing majority voting in the current implementation.

A classification module consists of two main phases: training and testing. General struc-

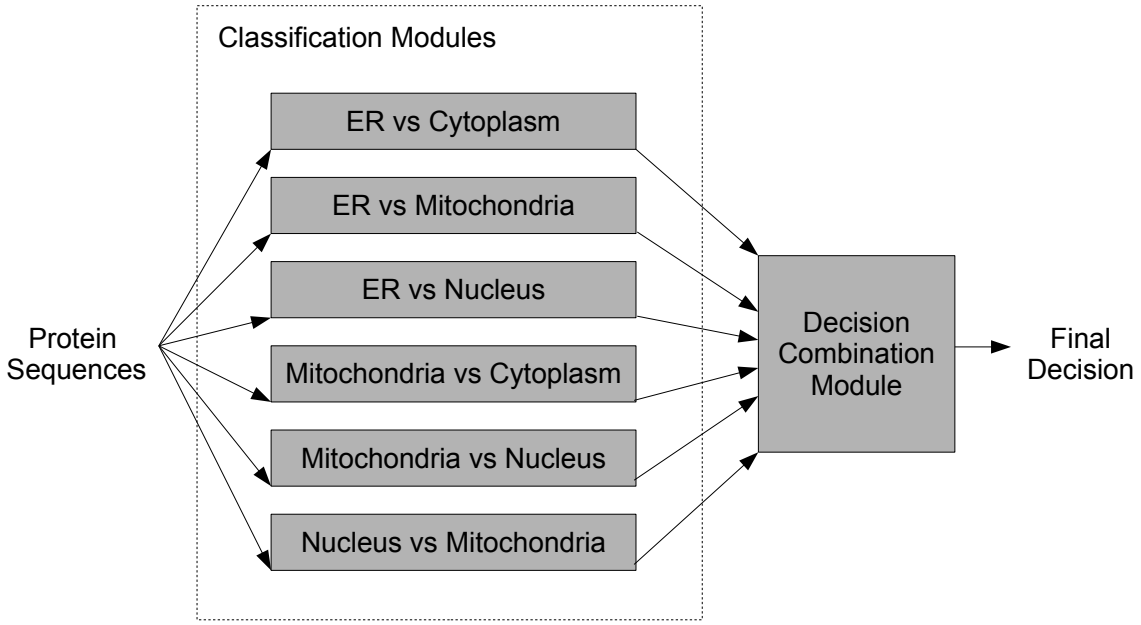


Figure 3.1: General system architecture.

ture of classification modules in training and testing phases are shown in Figure Figure 3.2 and Figure Figure 3.3. The training phase of classification modules basically consists of two main parts: feature extraction and classifier training. In feature extraction part, feature vectors are determined and they are used to train the classifier. Feature extractor module consists of three submodules: *Frequent Pattern Extraction*, *Feature Selection*, and *Feature Quantification*. Leaving the details of these submodules to the following sections, the basic flow of the training phase can be summarized as follows:

1. Training sequences are read.
2. Frequent patterns in the training dataset are identified.
3. Initial feature set is constructed by defining the features as the occurrences of the patterns in a protein sequence.

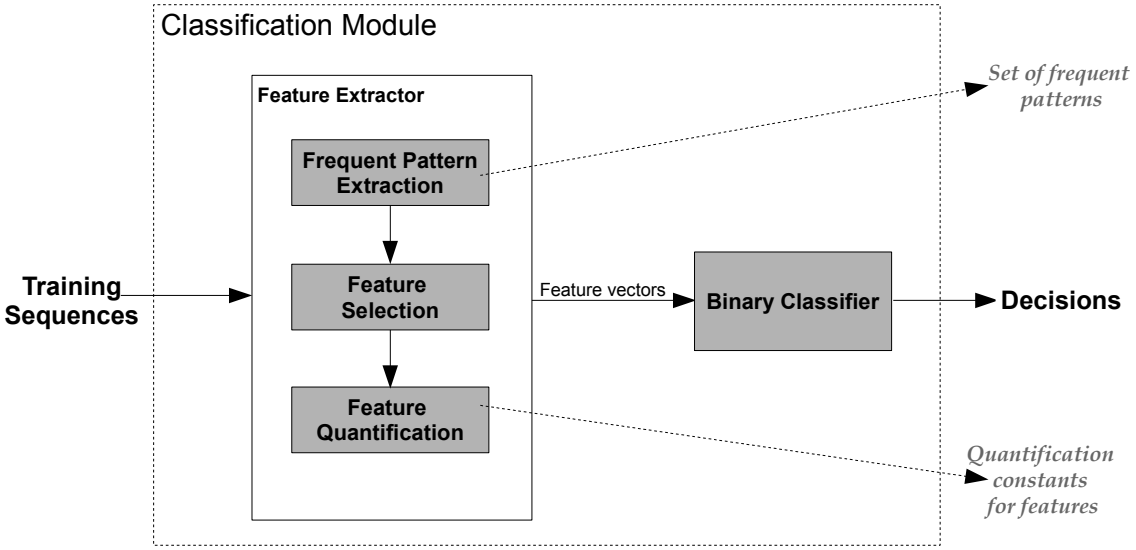


Figure 3.2: Classification module in training phase.

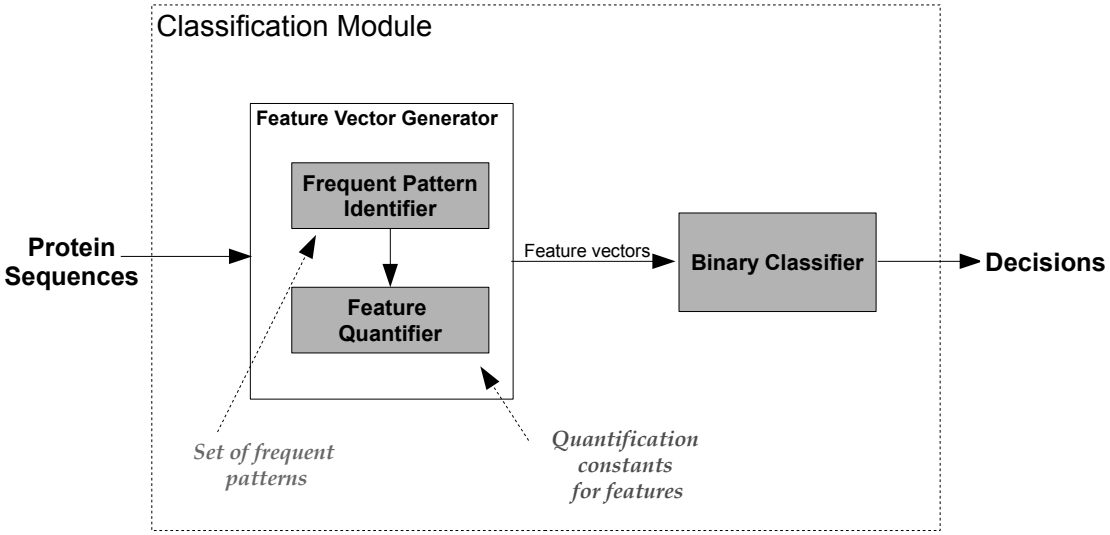


Figure 3.3: Classification module in testing phase.

4. A feature selection method is applied on the initial feature set. Selected features form the final feature set.

5. Features are quantified by using a feature quantification method and the feature vectors are formed.

6. Binary classifier is trained with the feature vectors corresponding to the training sequences.

In the training phase, some parameter values that are used in the testing phase are determined. These are *set of frequent patterns* and *quantification parameter for features*. These parameters are specific to each classification module.

In the testing phase, a classification module takes a protein sequence as input and outputs the predicted label of that sequence. Main submodules in this phase are *Feature Vector Generator* and *Binary Classifier*. Feature Vector Generator extracts features from the protein sequence, generates a feature vector and the binary classifier classifies into one of the two classes. Feature vector generator module contains submodules which are *Frequent Pattern Identifier* and *Feature Quantifier*. Frequent Pattern Identifier uses the *set of frequent patterns* which are determined in the training phase. It identifies the occurrences of these patterns in the protein sequence. Feature Quantifier quantifies the feature values using a frequency based metric which has parameters whose values determined in the training phase. After the frequent pattern identifier module generates the feature vector corresponding to the given protein sequence, Binary Classifier module outputs the predicted label of that sequence.

3.2 Modules

Main modules of the system are Classification Modules and Decision Combination Module. Combining decisions of binary classifiers is a widely used solution for handling multiclass classification problems. Decision Combination Module is basically for combining the decisions of binary classifiers, and it is based on majority voting strategy.

Classification Modules consists of two main phases: training and testing. In the training phase, the input is a training dataset $S = \{s_1, \ldots, s_m\}$ where m is the cardinality of the set and the corresponding label set is $C = \{c_1, \ldots, c_m\}$ where c_i is the label of the protein sequence s_i and $c_i \in C = \{+1, -1\}$. The outputs of the training phase are a set of frequent

patterns, quantification constants for features, and a trained binary classifier. The steps carried out in the training phase are as follows.

1. Frequent patterns are identified using level-wise string-mining algorithm [40]. Frequent pattern set is denoted by $P' = \{p_1, \ldots, p_n\}$ where n is the cardinality of the set.

2. Features are defined in a way that a feature f_i encodes the presence of a pattern p_i. Initial feature set is denoted by $F' = \{f_1, \ldots, f_n\}$.

3. A feature selection method is applied on F'. Selected features form the final feature set $F = \{f_1, \ldots, f_t\}$. The corresponding frequent pattern set is denoted by $P = \{p_1, \ldots, p_t\}$ where $t < n$.

4. A frequency based quantification method is used for determining the numerical values of the features for each sequence. A protein sequence s_i is mapped to a feature vector $v_i = \{v_{i1}, \ldots, v_{it}\}$ where v_{ij} denotes the value of f_j for protein sequence s_i.

5. The binary classifier is trained with the feature vectors $V = \{v_1, \ldots, v_m\}$.

The quantification method is based on the *term-frequency inverted document frequency* fTFIDF) [53] weighting scheme. By using TFIDF weight, features that appear more in a sequence are rewarded and that are common in all sequences are penalized. Training set is used for determining the commonness measure of features. This measure can be denoted by d and $D = \{d_1, \ldots, d_t\}$ is the set of these constants that is output at the end of the training phase.

At the end of the training phase, frequent pattern set $P = \{p_1, \ldots, p_t\}$ and quantification constant set $D = \{d_1, \ldots, d_t\}$ are determined and the classifier is trained. By taking these as inputs, Classification Module of the test phase begins. The following steps are carried out in the test phase:

1. Input sequence is examined to seek for the frequencies of patterns in the pattern set P.

2. Feature vector is constructed using the quantification method with the constant set D.

3. This feature vector is given to the already trained classifier to get a decision.

3.2.1 Finding frequent patterns

Our method uses frequent patterns of successive amino acids at the feature extraction stage. A level-wise search strategy [40] which is based on *a priori* property is used to find frequent

patterns in a protein dataset.*A priori* property is used by most frequent pattern mining algorithms. Its basic principle is stated as follows: If an itemset A is frequent, then every subset of A must be frequent. On the other hand, if an itemset A is infrequent, then any superset of A is also infrequent. All level-wise search algorithms use this property as a base. By using *a priori* property, search space is narrowed down level by level since the candidate itemset for the next level only grows from previous level's itemset [66].

For the formalization of the problem, some definitions and explanations are necessary. In the sequel, we follow the notation of von Birzele and Kramer [9].

Let Σ be the alphabet of the 20 amino acids with $\Sigma = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$. A protein sequence $s \in \Sigma^*$ is defined as any length string over the alphabet Σ and the protein dataset $S \subset \Sigma^*$ is a subset of all possible strings over Σ. Without expanding the alphabet with some amino acid groups that share common characteristics, the problem turns to be a simple text classification problem. Embedding of biological information is achieved by defining groups. Let G be a set of amino acid groups that share common chemical, structural or evolutionary properties. Group definitions are in similar syntax with a regular expression, e.g., [DR] means either D or R has to match the sequence. These are the defined groups, $G = \{$[HKR], [DE], [FYW], [VIL], [STDNGA], [DENQRS], [VILMFA], [EAL], [VIYWF], [PGND]$\}$ following the Taylor classification [67] with three additional groups of amino acids frequently found in α-helices ([EAL]), β-sheets([VIYWF]) and coil segments ([PGND]). Pattern alphabet is expanded by these groups, so the alphabet definition becomes, $A = \Sigma \cup G$ and a pattern is now defined over the alphabet p.

Our aim is to identify all patterns p in a protein dataset S that are frequent by looking at the minimal support *minSup* which is specified by the user. The frequency of a pattern in a dataset is defined as $freq(p, S) = \sum_{s \in S} number\ of\ occurrences\ of\ p\ in\ s$.

A pattern $p = p_1, p_2, \ldots, p_n$ can only be frequent if its two subpatterns $p_1, \ldots, p_{n-1}$ and $p_2, \ldots, p_n$ have already been found to be frequent. As we stated earlier, this holds because of the *a priori* property. Because of this fact, the algorithm searches the space in a level-wise manner and all the other parts of the search space that doesn't contain frequent subpatterns can be pruned. The pseudo code of the level-wise string-mining [40] algorithm is given in Figure Figure 3.4.

User initiates the process by providing the first level candidate frequent set (i.e. the alphabet elements), dataset and initial minimum support value. In the original form of the algorithm, minimum support value does not change throughout the search; but we observed in our experiments that relaxing the constraint as the level increases leads to better results.

```
Input: Initial first-level candidate pattern list P, data set S, initial minimum
       support value initialMinSup
Output: Frequent-pattern list
begin
    level ← 1;
    C_level ← {{p}|p ∈ P};
    F_all ← {};
    while C_level ≠ {} do
        // determine frequent candidates
        F_level ← {};
        CANDIDATE: forall candidates c ∈ C_level do
            occ ← 0;
            forall sequencess ∈ D do
                occ ← occ + #occurences of c in s;
                if (occ >= minSup) then
                    F_level ∪ c;
                    go to CANDIDATE;
                end
            end
        end
        // generate candidates for next level
        C_nextlevel ← {};
        index ←all frequent candidate prefixes of length (level-1) as keys and
          lists of their extensions as values;
        forall f ∈ F_level do
            suffix ←suffix of length (level-1) of f;
            if (index contains suffix) then
                extensions ← list of extensions for suffix from index;
                forall e ∈ extensions do
                    C_nextLevel∪concat(f, e);
                end
            end
        end
        level ← level + 1;
        F_all ∪ F_level;
    end
    return F_all ;
end
```

Figure 3.4: Pseudo code for level-wise string-mining algorithm (adopted from [9]).

This is because of the fact that having a longer common pattern is more difficult than having a short common pattern. So, the reduction on the value of the initial minimum support value helps on finding longer common patterns. Reduction begins after the fifth level and up to tenth level, initial value is reduced by twenty percent, and after the tenth level it is not changed anymore.

At each iteration of the algorithm, candidate pattern set is investigated and the ones that are frequent are added to the final frequent pattern set. In addition, the candidate pattern set for the next level is constructed by using the frequent patterns found in previous levels. Iterations continue until there are no more patterns in the candidate pattern set.

3.2.2 Feature quantification

The initial step of a learning algorithm is to define discriminative features. In this study, discriminative features are thought to be the occurrences of frequent patterns. The next step is feature quantification, which is defined as deciding how to assign a numerical value to every feature. A very basic approach will be assigning the feature values as the frequencies of the patterns in a given sequence. By this approach, each feature is taken to have the same importance. However, the commonness of a feature at a dataset can be used to evaluate the importance of the feature. Based on the *term-frequency inverted document frequency* (TFIDF)[53] weighting scheme, features that are common in all sequences are penalized and the ones appear more in a sequence are rewarded.
Given a pattern p,

- its corresponding feature is denoted by f,
- the total number of sequences in the training dataset S is defined as $\|S\|$,
- a particular feature vector for the sequence in the training dataset is denoted by F,
- TF_f denotes the number of occurrence of feature f in the feature vector F and
- SF_p denotes the number of feature vectors in S that contain pattern p at least once.

We can define the *TFIDF* value as follows:

$$\mathbf{TFIDF}(f) = \log(TF_f + 1.0) * \log(|S|/SF_p) \tag{3.1}$$

Here, TF_f is measuring how frequent the feature is in the sequence, and SF_p is measuring how common the feature is in the dataset. Clearly, $TFIDF(f)$ is proportional to TF_f, and

inversely proportional to SF_p values. $TFIDF$ value of every feature is calculated for each sequence.

$D = \{d_1, \ldots, d_t\}$ was defined as the set of quantification constants that are output at the end of the training phase. d_i is equal to $\log(|S|/SF_p)$ value of the feature f_i, the second multiplicand of the Equation 3.1. Since d_i values are independent of the sequence it is calculated for, and dependent only on the training dataset, they are accepted as the constants for Equation 3.1, and therefore calculated and fixed in the training phase.

3.2.3 Feature selection

Feature selection is the most important step of feature extraction. The method used for feature selection in this study is a filter method: *Pearson's chi-squared test* (independence test version). For finding an alternative filter method, a number of ranking methods are also implemented. Among likelihood measures *Poisson-stirling* which is the most widely used one among the likelihood measures is employed. Among asymtotic hypothesis tests, *t-test* and *likelihood ratio test* are implemented. *Pearson's chi-squared test* also belongs to the asymptotic hypothesis tests class. Besides the hypothesis tests, direct ("point") estimates of association strength coefficients are used. The two most common coefficients of association strength are the mu-value, μ and the odds ratio, θ. In addition to the $maximum-likelihood$ estimates of these two coefficients, *mutual information* (MI) and *log odds-ratio*, *dice* and *liddell* are implemented. Finally, from heuristic measures, MI^2 and MI^3 are decided to be used. After a set of experiments where best treshold values for each measure are determined, their classification accuracies are compared on a reduced set of data. The best among them, *t-test* is chosen to be used as an alternative filter method on the real dataset. Besides, we implemented the most popular multivariate filter method, Relief [35]. Among embedded methods, AdaBoost is employed.

Chi-squared test

Chi-squared test is the standard test for independence in 2-by-2 contingency tables. It is an asymptotic hypothesis test which uses the amount of evidence against the null hypothesis of independence as an association score. This amount can be quantified by the *p-value* of a statistical hypothesis test. Chi-squared test is based on a comparison of the observed frequencies with the expected frequencies under the point null hypothesis of independence. Chi-squared test is a sort of mean squared error formula which is scaled by the expected

variances of the frequencies. Chi-squared test statistic can be calculated from the values in Table Table 2.1 and Table Table 2.2 by the Equation 2.6.

The distribution of the chi-squared statistics is chi-square with $(r-1)*(c-1)$ degrees of freedom, where r is the row number and c is the column number of the contingency table. After determining the degrees of freedom, $p-value$ must also be determined for getting meaningful results. It is the probability of observing a value at least as extreme as the test statistic for the chi-square distribution with $(r-1)*(c-1)$ degrees of freedom. In our experiments, we have tried popularly used $p-values$: 0.05, 0.01 and 0.001.

t-test

t-test is also an asymptotic hypothesis test that also uses the amount of evidence against the null hypothesis of independence as an association score. *t-test* is based on a comparison of the observed frequencies with the expected frequencies. *t-test* statistics can be calculated from the values in Table Table 2.1 and Table Table 2.2 by the Equation 2.5.

Relief Algorithm

Relief algorithm which is developed by Kira and Rendell [35] has some extensions defined by Kononenko [36]. ReliefF is one of these extensions. It is not limited to two class problems as Relief algorithm is, and it is more robust and can deal with incomplete and noisy data. The pseudo code of the ReliefF algorithm is given in Figure Figure 3.5.

In ReliefF algorithm, a set of feature vectors for the training instances and their labels is taken and the vector W which contains the estimations of the qualities of the features is output. Initially, all the weights are set to be zero. For the randomly selected instance s_i in the training set S, k nearest hits H_j, which are the k nearest instances that belong to the same class that s_i belongs are found. And also, for each class that s_i does not belong, k nearest misses M_j are found. Then, for each feature, the quality estimation $W[f]$ is updated by the given formula where the projections of the distances between s_i and H_j (for all j) on the feature f are subtracted from $W[f]$, and the projections of the distances between s_i and M_j (for all j) on the feature f are added to $W[f]$. If s_i and instances of H have nearly same values on the feature f and s_i and instances M have different values on the feature f, then the feature f will get a high weight at that iteration. The iterations continue up to m, where m is a user-defined parameter.

Input: for each training instance a vector of feature values and the class value
Output: the vector W of estimations of the qualities of features
begin
  $W[f] \leftarrow 0.0$;
  for $i \leftarrow 1$ **to** m **do**
    Randomly select an instance S_i;
    find k nearest hits H_j;
    foreach *class* $C \neq class(R_i)$ **do**
      from class C find k nearest misses $M_j(C)$;
    end
    for $f \leftarrow 1$ **to** t **do**
      $W[f] \leftarrow W[f] - \sum_{j=1}^{k} diff(f, S_i, H_j) +$
      $\sum_{C \neq class(S_i)} \left[\frac{P(C)}{1-P(class(S_i))} \sum_{j=1}^{k} diff(f, S_i, M_j(C)) \right]$
    end
  end
end

Figure 3.5: Pseudo code for ReliefF algorithm (adopted from [50]).

3.2.4 Classification

Support Vector Machines [61] are used at the classification stage of this study in most of the experiment settings. In other settings, Adaboost is used as an embedded feature selection method for comparison.

For the multiclass classification problem we have, binary classifiers are decided to be used in a one-versus-one setting. This setting is more favorable to one-against-all setting [26]. For every possible different pair of classes, there must be a classifier in a one-versus-one setting. Therefore, for the subcellular localization problem with 4 compartments, 6 classifiers need to be used to cover the possible pairs formed by 4 classes.

CHAPTER 4

RESULTS AND DISCUSSION

The proposed system for the prediction of subcellular localization is composed of mainly three parts: feature extraction, feature selection (optional) and classification. The described feature extraction technique is adapted from a protein secondary structure prediction system proposed by Birzele and Kramer [9]. In this system, feature extraction stage is based on finding frequent subsequences and in the feature selection stage, chi-squared test and precision-recall binning is used in combination. In the system we describe, feature extraction is also based on finding frequent subsequences, but for the feature selection stage, a number of feature selection strategies are tested on the top of the basic strategy including the one proposed by Birzele and Kramer which is a combination of two statistical tests. We also investigate the without feature selection condition as a reference. Support Vector Machines (SVM) are used as the classifier in all the experiment settings except the one where the Adaboost algorithm is applied as an embedded feature selection method. For SVM, SVMlight software [32] with a radial basis function (RBF) kernel is employed. For Adaboost, GML AdaBoost Matlab Toolbox [64] is preferred.

The dataset consists of 3115 ER targeted (ER) , 1780 cytoplasmic(C), 1148 mitochondrial (M) and 2225 nuclear (N) animal protein sequences derived from Swiss-Prot database. It is the same database used in P2SL tool [4]. For all the experiment setting except a single one, this dataset is used. For this exceptional setting, where the ranking methods are compared against each other, a reduced set with 200 randomly selected sequences for each of the classes is used.

Conducted experiments can be grouped under six titles which are:

- Feature mapping without feature selection
- Feature selection with chi-squared and precision-recall binning tests

- Comparing ranking methods
- Feature selection with t-test
- Feature selection with Relief
- Feature selection with Adaboost

We performed a 4-fold cross validation while searching the best parameters for both the feature extraction and classification stages in all these experiments. Searched parameters are specific to the experiments and investigated in detail in the following sections. For each of the experiments classification accuracies in individual modules and confusion matrices representing the overall system performance are given. Results are presented as the average accuracy values of 4-fold cross-validation.

4.1 Feature mapping without feature selection

Searched parameters in this experiment setting are explained below.

- Level-wise string mining algorithm (see Table Figure 3.4) parameters:
  - Frequency threshold parameter *minSup*
- SVM parameters:
  - Trade-off between training error and margin, c
  - Gamma parameter in radial basis function (RBF) kernel, g

These parameter values differ in classifier modules. The best *minSup* values vary in the range 1500 to 3900. The best $\{c, g\}$ value pairs are $\{4, 0.01\}$ and $\{4, 0.001\}$. Individual module accuracies are given in Table Table 4.1. Overall system performance is represented by a confusion matrix which is given in Table 4.1. The overall accuracy of the system is 91.54% on the average (weighted on sequence numbers in each class) with a standard deviation of 0.24.

4.2 Feature selection with chi-squared test and precision-recall binning

Searched parameters in this experiment setting are explained below.

Table 4.1: Module accuracies with feature mapping without feature selection.

	Without Selection (%)
ER-C	96.04
ER-M	98.48
ER-N	96.06
M-C	95.35
M-N	96.50
N-C	91.31

Table 4.2: Confusion matrix for using the feature mapping method without feature selection.

Actual	Predicted (%)			
	C	ER	M	N
C	**88.26**	3.54	1.29	6.91
ER	2.98	**94.09**	0.35	2.58
M	7.58	2.00	**89.02**	1.39
N	7.39	2.81	0.90	**88.90**

- Level-wise string mining algorithm (see Table Figure 3.4) parameters:
  - Frequency threshold parameter $minSup$
- chi-squared test
  - p-value, ρ
- Precision-recall binning
  - Total bin number, bin
  - Precision percentage parameter, $precTresh$
- SVM parameters:
  - Trade-off between training error and margin, c
  - Gamma parameter in radial basis function (RBF) kernel, g

These parameter values differ in classifier modules. The best $minSup$ values change in the range 1500 to 3900. The best $\{c, g\}$ value pairs are $\{4, 0.01\}$ and $\{4, 0.001\}$.The best ρ values are $0.05, 0.01$. The total bin number is chosen to be 100 and the precision percentage parameter is determined to be 70. Overall system performance is represented by a confusion matrix which is given in Table Table 4.3.

Table 4.3: Confusion matrix for using the feature mapping method with chi-squared test and precision-recall binning.

Actual	Predicted (%)			
	C	ER	M	N
C	**87.13**	3.88	1.91	7.08
ER	2.84	**93.76**	0.56	2.84
M	6.44	2.70	**89.37**	1.48
N	7.78	2.70	1.15	**88.31**

Table 4.4: Module accuracies with chi-squared test.

	chi-squared (%)
ER-C	95.94
ER-M	98.50
ER-N	96.06
M-C	95.39
M-N	96.45
N-C	91.40

The overall accuracy of the system is 91.10% on the average with a standard deviation of 0.37. Since the overall accuracy is below the one achieved with the feature mapping technique without feature selection, we decided to change the feature selection strategy. Chi-squared test and precision-recall binning are applied individually, not subsequently. Chi-squared test, this time, improved the accuracies that are achieved with the feature mapping technique without feature selection, but precision-recall binning test did not improve the results. The results obtained by using chi-squared test are given are given in Table Table 4.4 and Table Table 4.5. The overall accuracy of the system is 91.71% on the average (weighted on sequence numbers in each class) with a standard deviation of 0.4.

Table 4.5: Confusion matrix for the chi-squared test.

Actual	Predicted (%)			
	C	ER	M	N
C	**87.92**	3.31	1.97	6.80
ER	2.60	**94.10**	0.42	2.88
M	6.01	1.92	**90.51**	1.57
N	6.99	2.67	1.09	**89.24**

Table 4.6: Module accuracies of ranking methods.

	ER-C	ER-M	ER-N	M-C	M-N	N-C (%)
MI	**85.75**	**91.12**	**86.12**	81.37	91.5	83.87
Odds Ratio	85.5	90.87	84.87	80.25	90.62	**84.37**
Dice	84.87	90.5	**86.12**	80.87	90.62	**84.37**
Poisson Stirling	85.62	91	85.87	***81.62***	91.5	84.12
T-test	**85.87**	***91.25***	***86.25***	**81.5**	**91.62**	83.87
Log-likelihood	84.75	90.37	85.37	**81.5**	***92***	83.75
Liddell	***86.12***	**91.12**	***86.25***	81.25	***92***	84.12
MI^2	84.62	90.5	**86.12**	80.87	90.75	***84.87***
MI^3	84.5	90.12	**86.12**	80.87	91	***84.37***

4.3 Comparing ranking methods

Searched parameters in this experiment setting are explained below.

- Level-wise string mining algorithm (see Table Figure 3.4) parameters:
  - Frequency threshold parameter *minSup*
- Ranking method
  - Percentage parameter, *percent*
- SVM parameters:
  - Trade-off between training error and margin, c
  - Gamma parameter in radial basis function (RBF) kernel, g

These parameter values differ in classifier modules. The best *minSup* values change in the range 245 to 315. The best *percent* values change in the range 50 to 90. The best {c, g} value pair is {4, 0.01}.

Individual module accuracies are given in Table Table 4.6. The italic ones are best accuracies in the column and the bold ones are in the best three. Prediction accuracies of the classes and the overall accuracy are given in Table Table 4.7 for all methods. The bold ones are best accuracies in the column.

As it is observed in Table Table 4.7, t-test achieves the best accuracy among the other methods. Therefore, we decided to conduct an experiment with t-test in the real dataset.

Table 4.7: Prediction accuracies of ranking methods.

	C	ER	M	N	Overall (%)
MI	68.75	76.25	71.75	77	75.08
Odds Ratio	69.25	77.5	71.75	75.5	75.41
Dice	66.5	78.75	**72.75**	77.25	76.25
Poisson Stirling	**70.25**	76.25	72.25	**77.5**	75.45
T-test	69.75	**79.25**	**72.75**	**77.5**	**77**
Log-likelihood	69.5	75.25	71.75	76.5	74.53
Liddell	70	76	72.5	**77.5**	75.31
MI^2	68.5	78.25	72	77.25	76.18
MI^3	68	77.25	72.25	76.5	75.42

4.4 Feature selection with t-test

Searched parameters in this experiment setting are explained below.

- Level-wise string mining algorithm (Table Figure 3.4) parameters:
  - Frequency threshold parameter *minSup*
- T-test
  - Threshold parameter, *tresh*
- SVM parameters:
  - Trade-off between training error and margin, c
  - Gamma parameter in radial basis function (RBF) kernel, g

These parameter values differ in classifier modules. The best *minSup* values change in the range 1500 to 3900. The best $\{c, g\}$ value pairs are {4, 0.01} and {4, 0.001}. The best *tresh* values change in the range 0.4 to 1.6. Individual module accuracies are given in Table 4.8. Overall system performance is represented by a confusion matrix which is given in Table 4.4. The overall accuracy of the system is 91.39% on the average (weighted on sequence numbers in each class) with a standard deviation of 0.35.

4.5 Feature selection with Relief algorithm

Searched parameters in this experiment setting are explained below.

- Level-wise string mining algorithm (see Table Figure 3.4) parameters:

Table 4.8: Module accuracies with the t-test.

	t-test (%)
ER-C	95.93
ER-M	98.53
ER-N	95.73
M-C	94.94
M-N	96.60
N-C	90.99

Table 4.9: Confusion matrix for the t-test.

Actual	Predicted (%)			
	C	ER	M	N
C	**87.70**	3.31	1.85	7.13
ER	2.70	**93.81**	0.43	3.06
M	6.44	2.00	**90.07**	1.48
N	7.44	2.58	1.15	**88.82**

  - Frequency threshold parameter *minSup*

- Relief algorithm parameters
  - Number of nearest neighborhoods, k
  - Percentage parameter, *percent*
- SVM parameters:
  - Trade-off between training error and margin, c
  - Gamma parameter in radial basis function (RBF) kernel, g

The parameter k is chosen to be 10. The best *minSup* values change in the range 1400 to 3900. The best *percent* values change in the range 86 to 96. The best $\{c, g\}$ value pairs are $\{4, 0.01\}$ and $\{4, 0.001\}$. Individual module accuracies are given in Table Table 4.10. Overall system performance is represented by a confusion matrix which is given in Table 4.5. The overall accuracy of the system is 91.31% on the average (weighted on sequence numbers in each class) with a standard deviation of 0.23.

4.6 Feature selection with Adaboost algorithm

Searched parameters in this experiment setting are explained below.

Table 4.10: Module accuracies with the Relief algorithm.

	Relief (%)
ER-C	95.79
ER-M	98.32
ER-N	95.94
M-C	95.35
M-N	96.62
N-C	91.24

Table 4.11: Confusion matrix for the Relief algorithm.

Actual	Predicted (%)			
	C	ER	M	N
C	**87.81**	3.43	1.52	7.25
ER	2.85	**93.64**	0.56	2.95
M	6.71	2.35	**89.55**	1.39
N	7.39	2.61	1.04	**88.96**

- Level-wise string mining algorithm (see Table Figure 3.4) parameters:
  - Frequency threshold parameter *minSup*
- Adaboost algorithm parameters
  - Number of maximum splits of the tree learner, *maxSplit*
  - Number of maximum iterations, *maxIter*
- SVM parameters:
  - Trade-off between training error and margin, c
  - Gamma parameter in radial basis function (RBF) kernel, g

A tree learner with 3 maximum splits is used as the weak learner. The best *maxIter* values change in the range 270 to 390. The best *minSup* values change in the range 1500 to 4000. The best $\{c, g\}$ value pairs are $\{4, 0.01\}$ and $\{4, 0.001\}$. Individual module accuracies are given in Table Table 4.12. Overall system performance is represented by a confusion matrix which is given in Table Table 4.13. The overall accuracy of the system is 90.60% on the average (weighted on sequence numbers in each class) with a standard deviation of 0.26.

Table 4.12: Module accuracies with the Adaboost algorithm.

	Adaboost (%)
ER-C	94.21
ER-M	97.59
ER-N	95.11
M-C	93.92
M-N	96.26
N-C	90.28

Table 4.13: Confusion matrix for the Adaboost algorithm.

Actual	Predicted (%)			
	C	ER	M	N
C	**87.36**	2.81	2.30	7.53
ER	3.78	**92.54**	0.88	2.79
M	6.27	1.92	**90.16**	1.65
N	7.75	2.61	1.15	**88.48**

4.7 Comparison of the results of the algorithm

For comparison, individual module accuracies for all methods are given in Table Table 4.14. The bold accuracies are in best two. The prediction accuracies of the classes and the overall accuracy are given in Table Table 4.15 for all methods. As it is observed from Table Table 4.15, chi-squared test achieves the best accuracies among the other feature selection methods. For the method with chi-squared test, precision and recall values for each class are given in Table Table 4.16

In addition to confusion matrix for each class the evaluation of the P2SL prediction results was presented by four statistical measures defined as: precision, recall, F-score and specificity.

Table 4.14: Individual classifier accuracies for all methods.

	Without Selection	Chi-squared	T-Test	Relief	Adaboost
ER-C	**96.04**	**95.94**	95.93	95.79	94.21
ER-M	98.49	**98.5**	**98.53**	98.32	97.59
ER-N	**96.06**	**96.06**	95.73	**95.94**	95.11
M-C	**95.35**	**95.39**	94.94	**95.35**	93.92
M-N	96.50	96.45	**96.60**	**96.62**	96.26
N-C	**91.31**	**91.40**	90.99	91.24	90.28

Table 4.15: Total prediction accuracies of methods.

	Without Selection	Chi-squared	T-Test	Relief	Adaboost
C	**88.26**	87.92	87.70	87.81	87.36
ER	94.09	**94.10**	93.81	93.64	92.54
M	89.02	**90.51**	90.07	89.55	90.16
N	88.90	**89.24**	88.82	88.96	88.48
overall	91.54	**91.71**	91.39	91.31	90.60

Table 4.16: Precision and recall rates for chi-squared method.

	C	ER	M	N (%)
precision	84.93	92.25	96.30	88.80
recall	87.92	94.10	90.51	89.24

4.8 Comparison of the results with other systems

We compared the results obtained with the proposed method with P2SL [4]. P2SL is also a subcellular localization prediction tool which is designed for the same 4 classes in our system. The results for both systems are given in Table Table 4.17.

We also compared the results obtained with the proposed method with TargetP [44]. TargetP is a subcellular localization prediction tool which is based on neural networks. It is designed for 3-class classification where classes are ER, Mitochondria and Other. The results for both systems are given in Table Table 4.18. In terms of training and testing execution times, our method is comparable with P2SL and TargetP in the order of magnitude.

Table 4.17: Comparison of P2SL and our system.

Actual	Predicted (%)				
		C	ER	M	N
C	*Our Sys.*	**87.92**	3.31	1.97	6.80
	P2SL	**79.68**	3.81	3.56	12.93
ER	*Our Sys.*	2.60	**94.10**	0.42	2.88
	P2SL	8.21	**83.97**	3.99	3.81
M	*Our Sys.*	6.01	1.92	**90.51**	1.57
	P2SL	16.02	6.59	**75.45**	1.93
N	*Our Sys.*	6.99	2.67	1.09	**89.24**
	P2SL	30.10	3.51	3.10	**63.27**

Table 4.18: Comparison of TargetP and our system.

Actual	Predicted (%)			
		ER	M	Other
ER	*Our Sys.*	**94.10**	0.42	5.44
	TargetP	**85.21**	2.06	12.73
M	*Our Sys.*	1.92	**90.51**	7.58
	TargetP	3.64	**77.84**	18.52
Other	*Our Sys.*	5.98	3.06	**90.96**
	TargetP	1.88	8.86	**89.27**

Table 4.19: Number of features used in the modules for all methods.

	Without Selection	Chi-squared	T-test	Adaboost	Relief
ER-C	2746	2738	2559	930	3089
ER-M	2583	2574	2060	1170	2557
ER-N	8066	7405	7592	900	8519
M-C	5383	2925	1809	810	4730
M-N	5791	3914	2716	960	6920
N-C	13538	9678	7379	840	12267

4.9 Investigation of frequent patterns

Number of features used in modules depend on the feature selection algorithm employed and are given in Table Table 4.19 for all methods averaged on for 4 folds. These are also the number of frequent patterns. Frequent pattern distribution of a module is an interesting information and it is given in Table Table 4.20 for the case where no feature selection algorithm is applied.

If frequent patterns are ranked over their effect on classification performance, then top ranked ones can be examined for finding a biological meaning under the classification performance of the method. However, in the method we describe, support vector machines (SVM)

Table 4.20: Pattern length distributions of modules.

Module	Pattern Length								
	2	3	4	5	6	7	8	9	10
ER-C	435	1243	1014	23					
ER-M	433	1182	856	81					
ER-N	621	2801	3339	1212	60	2			
M-C	550	2173	2197	433					
M-N	563	2234	2271	653	38	1			
N-C	686	3739	5694	2738	588	34	13	14	1

Table 4.21: Partial coverage of patterns for the 3rd motif of Chymotrypsin fingerprint

Pattern	Partial coverage
$[STDNGA]C$	KD..KGDSGGPLI
$[PGND][PGND][STDNGA][STDNGA]$	KDSCK....GPLI
$G[STDNGA]$	KDSCK..S..PLI
$P[VILMFA][VILMFA]$	KDSCKGDSGG...
$[STDNGA][STDNGA][HKR][STDNGA]$	..SCKGDSGGPLI
combination of all patterns	....K........

are used and in SVM classification, the learned decision functions are often difficult for people to understand and extract further biological information other than the prediction result. Being a rule based classification algorithm, Adaboost results are more understandable and readable compared to SVM. Therefore, we decided to analyze Adaboost results. Features are ranked over their effect on classification performance for each classification module and the patterns corresponding to these features are tried to be matched with known biological motifs. Since the average length of the patterns, which is about 3-4, are short compared to known motifs we think that a number of patterns may cover a known motif in combination. PRINTS database's FPScan tool is used for the analysis. Top ranked patterns are marked on a given sequence and marked fingerprints by FPScan tool on the same sequence are examined for assessing the coverage percent. Here is an example protein sequence on which 3 motifs of top ranked fingerprint (which is Chymotrypsin) are shown in bold.

MVLIRVLANLLILQLSYAQKSSELVIGGDECNINEHRSLV
VLFNSSGVLC**GGTLINQEYVLTAAHC**DMPNMQILLGVHSA
SVLNDDEQARDPEEKYFCLSSNNDT**EWDKDIMLIRLNRSV**
NNSVHIAPLSLPSSPPRLGSVCRVMGWGAITSPNETYPDV
PHCANINILRYSLCRAVYLGMPVQSRILCAGILRGG**KDSC**
KGDSGGPLICNGQLQGIVSAGSDPCAKPRVPNLYIKVFDY
TDWIQSIIAGNTTVTCPQ

In Table Table 4.21 partial coverage of top ranked patterns are given for the 3rd motif of Chymotrypsin fingerprint. Patterns are in regular expression format and partially covered area of the fingerprint is shown with dots. As it is observed from the last row, combination of the given patterns covers the 92% of the original motif.

Although true positive rates are high, because of the very high false positive rates, the

results are not seen meaningful. Due to the extended alphabet definition, the patterns we found are matching with so many subsequences of a protein sequence which leads to the fact that not only the fingerprints marked by FPScan tool, but almost whole sequence is covered by the patterns.

4.10 Discussion

Considering the results presented in this chapter, we see that:

- Feature selection methods do not help to increase the classification accuracy of ER-C and ER-N modules. For the other modules, employing feature selection methods leads to increases in classification accuracies.
- Among feature selection methods, Adaboost is the one that uses the least number of features. In addition, it is the one that has the lowest classification accuracy.
- For ER-C and ER-M modules, all the feature selection methods use about 2000-3000 features, except Adaboost with nearly 1000 features. For ER-N module, average number of features that feature selection methods use is about 8000. Adaboost use 900 features for ER-N module for getting the best accuracy. N-C module is the one where the highest number of features must be used for achieving better classification rates. Feature selection methods are more successful in M-C and M-N modules compared to other modules. In these modules, early half of the initial feature set is pruned by feature selection methods with a 0.1 percent gain of classification accuracy.
- In terms of execution times, Adaboost is the slowest one in training and the fastest one in testing.
- System can classify the ER targeted proteins correctly at higher rates. The lowest accuracies are on cytoplasmic and nuclear proteins and it is because of the fact that they are the ones mostly confused with each other.
- For all the classification compartments, our system outperformed TargetP in terms of classification rates. For ER targeted and mitochondrial proteins, our system increased the classification rates of TargetP by nearly 10 percent. For Other class, 1.7 percent increase is gained.

- Our system has better classification rates for all 4 compartments compared to P2SL. In the overall performance, our method improved the P2SL classification rate which is 81.96 by nearly 10 percent.
- In terms of training and testing execution times, our method is comparable with P2SL and TargetP in the order of magnitude.

CHAPTER 5

DESIGN AND IMPLEMENTATION OF THE CYTOSCAPE PLUG-IN

We design and implement a plug-in for the Cytoscape tool, named *PlugP2SL*. It annotates biological networks visualized in Cytoscape with subcellular localization predictions performed by Prediction of protein subcellular localization (P2SL) [4]. The main functionality of the system is to retrieve subcellular localization predictions of the selected nodes and to add a number of attributes to the nodes related with localization prediction. Attribute addition helps on the analysis of protein/gene networks, since Cytoscape lets users to define a mapping from data attributes to visual attributes like color, shape, and size.

A client-server architecture is established based on Simple Object Access Protocol (SOAP) technology with the Cytoscape plug-in *PlugP2SL* which is working as client. On the server side, there is Model Organisms Proteome Subcellular Localization Database (MEP2SL) [5] database which stores subcellular localization information of nine model organisms which are predicted by P2SL tool. For the other proteins that do not belong to one of the nine organisms collected in MEP2SL, there is also on demand prediction possibility. General system flow is shown in Figure Figure 5.1.

User must provide gene name and organism name related to the selected protein in order to use *PlugP2SL*. As it is observed in Figure Figure 5.1, subcellular localization predictor, which is P2SL in the current system, demands sequence information and MEP2SL database demands UniRef100 id. In order to obtain the association between these attributes UniprotKB and UniRef100 databases are used for constructing an internal database. Before going into design and implementation details of rhe system, some background information is provided in the following section.

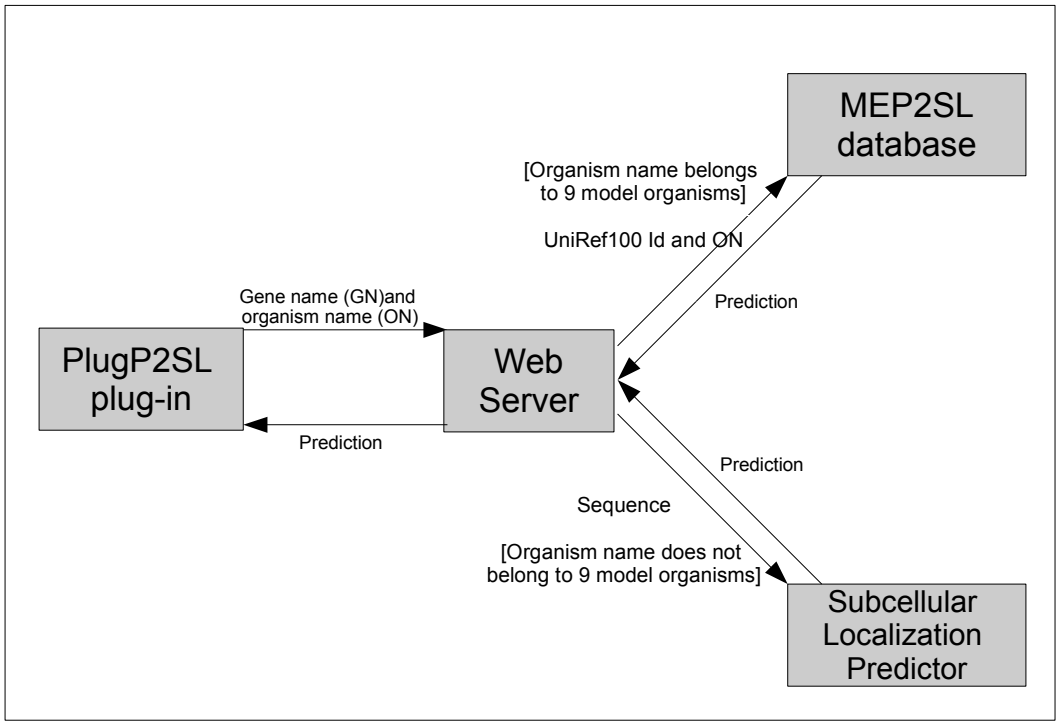


Figure 5.1: General system flow for *PlugP2SL* plug-in.

5.1 Background information

5.1.1 The Cytoscape tool and plug-in development

Cytoscape [47] is an open-source software program for network visualization and analysis. It provides basic functionality to visualize and query biomolecular interaction networks and to integrate them with any kind of annotations. The most important feature of the Cytoscape tool is its extendibility through the plug-in architecture constructed under the core application. There are lots of plug-ins [1] that are available for analyzing existing networks, importing networks and attributes in different file formats, inferring new networks, connecting with databases and functional enrichment of networks.

A typical biomolecular interaction network contains genes, proteins, some other biomolecules and interactions defined between these elements. Cytoscape uses nodes for representing network molecules and edges for representing interactions between them. It stores interaction graphs in two simple ASCII text formats, Simple Interaction File (SIF) and Graph Markup Language (GML). Each line in a SIF specifies a source node, a relationship type, and one or more target nodes. Some common relationship types are protein-protein interaction (pp),

protein-DNA interaction (pd), etc. GML file stores the interaction data with the information about how to view it such as layout, colors, etc.

Cytoscape allows user to integrate arbitrary information with interaction networks by adding attributes to nodes, to edges or to the network. User is allowed to define a mapping from data attributes to visual attributes like colors, shapes [47]. By this way, integrated data is used for analyzing the network. Adding attributes to nodes and edges can be done by importing node and edge attribute files.

5.1.2 UniProt

Universal Protein Resource (UniProt) [3] is a central repository of protein data. It is a freely accessible, centralized resource for protein sequences and functional information. It is originated from merging three databases: Swiss-Prot, TrEMBL, and PIR. UniProt is composed of three components: UniProt Knowledgebase (UniProtKB), UniProt Archive (UniParc), and UniProt Reference Clusters (UniRef).

UniProtKB constitutes the core of the UniProt and contains expertly curated protein information, including function annotations, cross-references to multiple sources [12]. UniProtKB consists of two sections, which are UniProtKB/Swiss-Prot and UniProtKB/TrEMBL. The former contains manually annotated records with information extracted from literature and computational analysis while the latter contains records that are computationally analyzed by classification and automatic annotation.

UniParc is a comprehensive non-redundant protein sequence collection that includes historical information on all protein sequences [38]. Protein sequences are revised daily from many publicly available sources, not only the UniProt Consortium databases Swiss-Prot, TrEMBL and PIR-PSD, but also EMBL, Ensembl, IPI, PDB, RefSeq, FlyBase, WormBase, Patent Offices, etc. UniParc assigns a unique UniParc identifier to each unique sequence. Furthermore cross-references to the source databases with source accession numbers and sequence versions are provided.

UniRef databases combine closely related sequences based on sequence identity to speed up searches. They provide three clustered sets of sequences from UniProtKB and selected UniParc records: UniRef100, UniRef90, and UniRef50. The UniRef100 database combines identical sequences into a single UniRef entry, which contains representative protein's sequence information, accession numbers of the UniProtKB entries and links to the UniProtKB and UniParc records that constitutes the UniRef entry. UniRef90 and UniRef50 are built by clustering UniRef100 sequences. UniRef90 contains clusters in which sequences have

at least 90 percent sequence identity to the representative sequence [12]. UniRef50 is like UniRef90 with a constraint of 50 percent sequence identity.

UniProtKB/Swiss-Prot database

Each UniProtKB/Swiss-Prot database entry consists of elements which are: accession, name, protein, gene, organism, organismHost, geneLocation, reference, comment, dbReference, keyword, feature, evidence, sequence and dataset [41]. In this study, we are interested in accession, gene, organism and sequence elements.

Accession numbers provide a stable way of identifying entries from release to release. Entries will have more than one accession number if they have been merged or split. The first accession number is the primary accession number, and it is the one used when the entry is referenced from other database sources.

Gene element contains the name(s) of the gene(s) that code for the stored protein sequence. A gene name can be in one of the following types: primary, synonym, ordered locus and ORF. Primary gene names are the major concern in this work. Entries can have more than one gene element.

Organism element describes the source of the stored protein sequence. An organism name can be in one of the following types: common, full, scientific, synonym, abbreviation. Organism names in scientific type are the major concern in this work.

Sequence element basically contains the stored sequence information.

UniRef100 database

Each UniRef100 database entry consists of elements which are: name, property, representative member and member. There is only one representative member in an entry, but there are a number of members. Representative member and member elements contain references to databases UniProtKB, UniParc, UniRef90, UniRef50. They also contain source organism name and sequence information.

5.1.3 Model Organisms Proteome Subcellular Localization Database (MEP2SL)

MEP2SL is an automatically updated, downloadable, and searchable database which contains protein localization distribution information of *H. sapiens* and other eight model organisms from UniRef100 database [5]. Protein localization distribution information is obtained by prediction using P2SL tool. P2SL is a hybrid computational system that predicts over

ER targeted, cytosolic, mitochondrial and nuclear protein localization classes. Section 5.1.4 gives more detailed information on P2SL tool. In MEP2SL, localization distribution is cross referenced to UniProt and NCBI to get additional features and homologous sequences.

MEP2SL database contains the following eukaryotic organisms:

- Homo sapiens (Human)
- Mus musculus (Mouse)
- Rattus norvegicus (Rat)
- Drosophila melanogaster (Fruit fly)
- Brachydanio rerio (Zebrafish) (Danio rerio)
- Saccharomyces cerevisiae (Baker's yeast)
- Xenopus tropicalis (Western clawed frog) (Silurana tropicalis)
- Dictyostelium discoideum (Slime mold)
- Caenorhabditis elegans

For the construction of the MEP2SL database, UniRef100 database is automatically downloaded and parsed for identifying the proteins from the concerned organisms and achieving the data about proteins including sequence information and UniRef100 id. Then using P2SL tool, which utilizes protein sequence information for prediction, protein localization distribution information is associated with each protein.

MEP2SL database offers the distribution potential of a protein into more than one compartment; therefore multi-compartmental proteins are predicted in addition to their localization classes. Hence, it provides a reference source for eukaryotic proteome scale subcellular localization information.

5.1.4 Prediction of protein subcellular localization (P2SL)

P2SL is a system that predicts the subcellular localization of proteins in eukaryotic organisms based on implicit motif frequency distribution of protein sequences [4]. Self-organizing maps (SOM) are used for feature extraction and the boundaries among the classes are determined by a set of support vector machines (SVM). P2SL is a hybrid computational system that predicts over ER targeted, cytosolic, mitochondrial and nuclear protein localization classes.

P2SL uses protein sequence information as input and outputs one or more subcellular localization(s) as prediction with the computed possibilities associated to these localizations.

5.1.5 Web services

The official definition of the term "Web Service" which is specified by the World Wide Web Consortium (W3C) whose aim is to create standards and specifications for the web-related technologies, is as follows [14]:

"A Web service is a software system identified by a URI whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols."

A web service is usually identified by a Unified Recourse Identifier (URI) which is a formatted string that represents the address or location of resources available on the Internet. A web service has Web Service Description Language (WSDL) definitions. A WSDL document is written in XML, and it basically describes a web service by specifying the location of the service and the methods service provides. To communicate with web services we need to use SOAP messages, which are XML based messages transported over Internet protocols like HTTP, SMTP, and FTP [69]. Web services can be better described with the Figure Figure 5.2. Here, the service requester is the client of the web service, and the service provider is the host of the web service. A requester makes requests to the provider and the provider responses accordingly. A service registry is a sort of database of web services which contains definitions and URIs for web services. Universal Description, Discovery, and Integration (UDDI) defines the standards for registries on storing and publishing descriptions of network services in terms of XML messages [69]. A web service developer usually publishes the WSDL for the constructed web services in the registry. A client can query the registry using UDDI and get the WSDL from the registry.

Web services are generally used when there is a need to access the functionality provided by a remote server through the Internet. They can be seen as an abstraction layer positioned between the application code which serves the needed functionality to the user and the user of that code, the client. By the help of web services, consumers and providers can easily connect in a language-independent and platform-independent way [6]. The three core web services standards and technologies for building and enabling web services are SOAP, WSDL and UDDI. An overview of each technology is presented in the following sections.

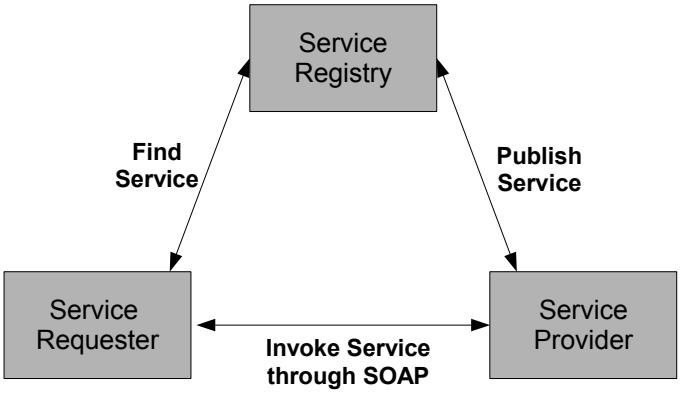


Figure 5.2: Web services model (adopted from [69]).

Simple Object Access Protocol (SOAP)

SOAP [65] is used for communication between applications through the Internet. It is a simple XML based envelope to transfer the information and a set of rules to translate the application and platform specific data types into the XML representation. XML message is shown in Figure Figure 5.3.

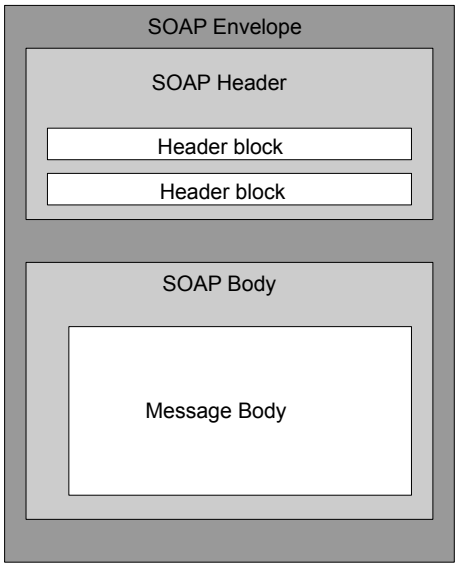


Figure 5.3: SOAP message model (adopted from [57]).

Each SOAP message has a mandatory SOAP envelope which consists of an optional

SOAP header, and a mandatory body. The envelope element may contain a header element, which must be the first element after the envelope element. The purpose of the header block is to communicate context specific information relevant for the processing of the SOAP message. The envelope must contain exactly one body element. The content of the body element is the original message [57].

Since XML document can be created on any platform with any operating system running on it and by any programming language, SOAP is platform and language independent. SOAP can be used in a variety of messaging systems and can be transported over a variety of Internet protocols like HTTP, SMTP, and FTP. In the web service architecture, it represents a cornerstone because of the fact that it enables diverse applications to easily exchange services and data by defining a way to perform remote procedure calls.

Web Service Description Language (WSDL)

WSDL is a specification which defines how to describe web services in a common XML grammar [31]. WSDL is used to describe what a web service can do, where it is located, how it is invoked. WSDL document uses the following elements in the definition of web services [62]:

- **Types:** A container for data type definitions using some type system (such as XSD).
- **Message:** An abstract, typed definitions of data being exchanged.
- **Operation:** An abstract description of an action supported by the service.
- **Port type:** An abstract set of operations supported by one or more endpoints.
- **Binding:** A concrete protocol and data format specification for a particular port type.
- **Port:** A single endpoint defined as a combination of a binding and a network address.
- **Service:** A collection of related endpoints.

Universal Description Discovery and Integration (UDDI)

UDDI defines the standards for registries on storing and publishing descriptions of web services in terms of XML messages [69]. It is a directory where web services can be registered and assigned to service providers. UDDI communicates via SOAP messages.

5.2 Design and implementation details of *PlugP2SL*

PlugP2SL is a Cytoscape plug-in which works as client in the constructed client-server architecture. Server side of this architecture is based on SOAP technology. On the server side, there is MEP2SL database which stores subcellular localization information of nine model organisms which are predicted by P2SL tool and P2SL predictor that is used for on demand prediction of proteins that do not belong to one of the nine organisms collected in MEP2SL. As it is observed in Figure Figure 5.4 which elaborates Figure Figure 5.1, internal database

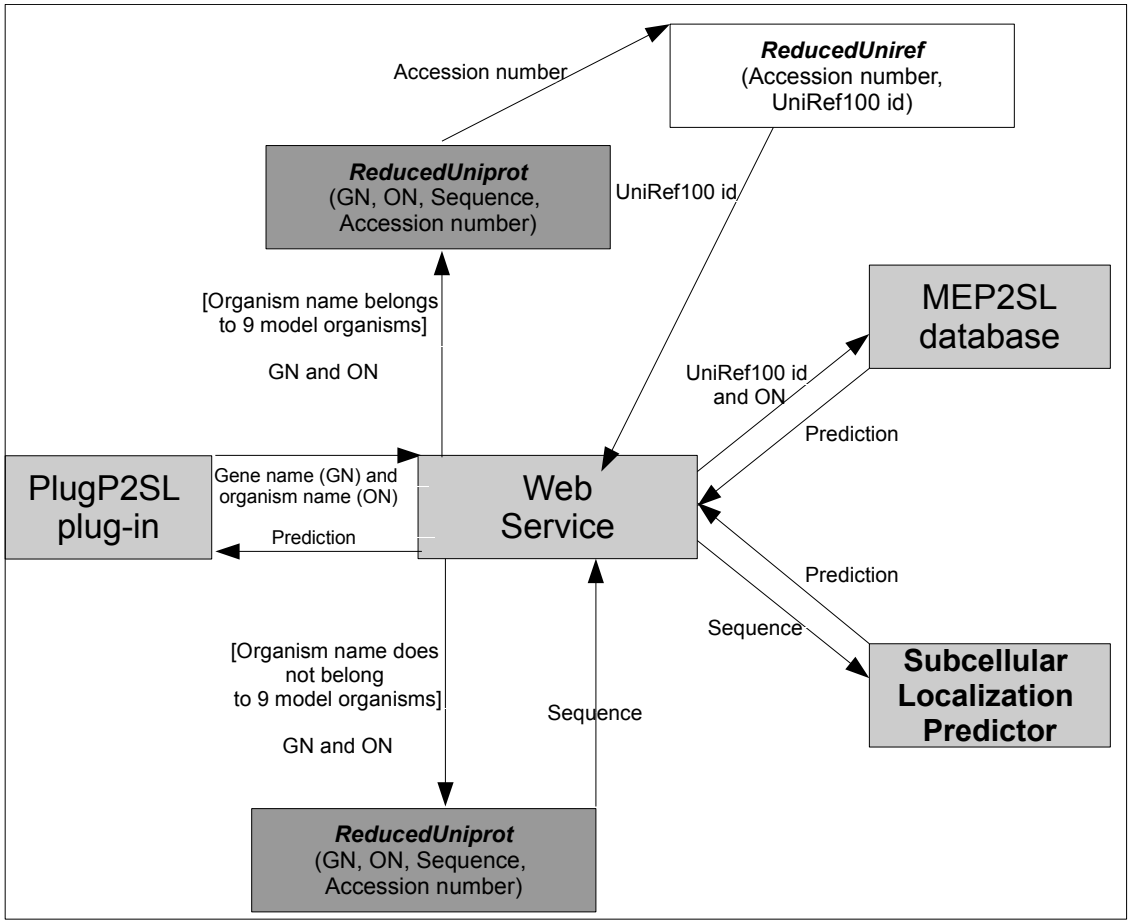


Figure 5.4: Detailed system flow for *PlugP2SL* plug-in.

tables *reducedUniprot* and *reducedUniRef* are used for the association problem of given and needed attributes. *UniprotKB* database is a central repository of protein data which contains various information about proteins, including gene name, organism name and sequence information. By downloading and parsing the UniprotKB database, a database table *reducedUniprot* is constructed in the internal database.

As it is explained in detail in Section 5.1.3, MEP2SL [5] is an automatically updated, downloadable, and searchable database which contains protein localization distribution information of H. sapiens and eight other model organisms from UniRef100 database. MEP2SL database can be queried with UniRef100 id to retrieve the subcellular localization prediction. UniRef100 is a database that combines identical sequences into a single entry, which contains representative protein's sequence information, accession numbers of the UniProtKB and UniParc entries and links to the UniProtKB and UniParc records that constitutes the UniRef entry. It is possible to reach UniRef100 id if the accession number of a UniProtKB entry is retained. Therefore, for associating the user inputs gene name and organism name with UniRef100 id, both UniprotKB and UniRef100 databases are needed. By downloading and parsing the UniRef100 database, a database table *reducedUniRef* is constructed in the internal database. Keeping the internal database up to date, a process is initiated that periodically updates the constructed tables.

For better understanding of the design elements of the *PlugP2SL* plug-in, Unified Modeling Language (UML) diagrams [55] are given in Section 5.3, Section 5.4, and Section 5.5.

5.3 Use cases

Use case diagram is given in Figure Figure 5.5. There are five actors defined in the system:

- User
- MEP2SL
- Predictor
- Timer

Retrieving subcellular localization predictions for the selected nodes is the main use case in the system. Retrieving organism list for the selected node (this use case is applicable for single node selections) is another use case in which the *PlugP2SL* user involved. Retrieving subcellular localization predictions can be done either querying the MEP2SL database, or using the P2SL tool. Internal database is updated periodically by downloding Uniprot databases; UniRef100 and UniProtKB/Swiss-Prot.

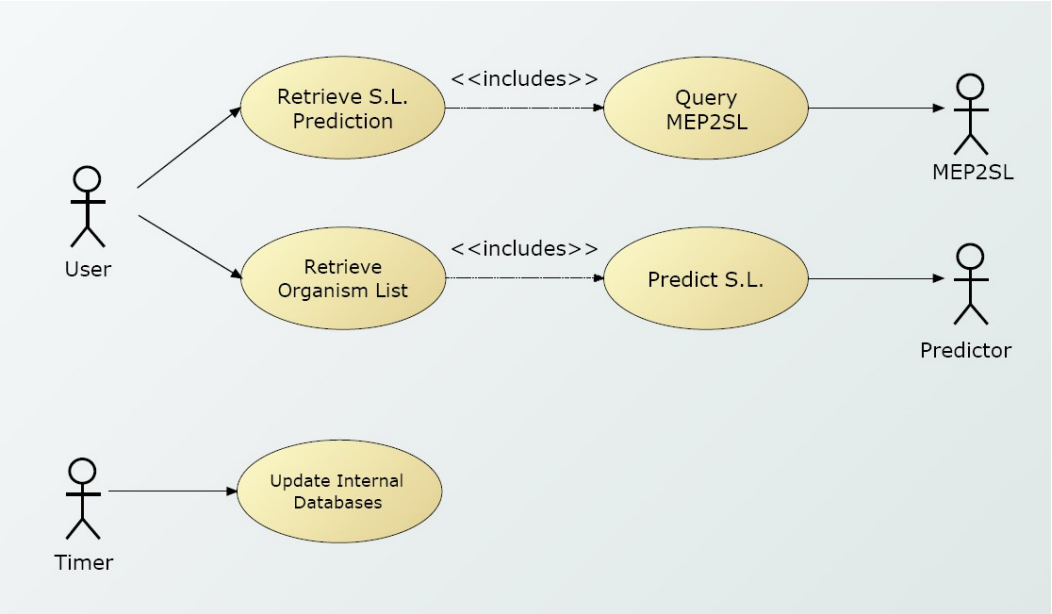


Figure 5.5: Use case diagram for *PlugP2SL*.

5.4 Sequence diagrams

5.4.1 Prediction request

After providing the gene name and the organism name to the plug-in, user initiates the process of getting the subcellular localization prediction. *PlugP2SL* calls the web service method *GetPrediction* with a string argument which is the concatenation of the gene name and the organism name separated with the special character ''.

At first, organism name is checked to see if it is one of the organisms MEP2SL database contains. If it is, then *reducedUniprot* database is queried to find the accession number corresponding to the given gene name and organism name. Subsequently, *reducedUniRef* database is queried to find the UniRef100 id corresponding to the given accession number and organism name. After UniRef100 id is reached, MEP2SL database can be queried to get the prediction.

If the organism name does not belong to MEP2SL organism list, then P2SL tool must be used. Details of the prediction stage are left to the next section. User must confirm to continue before the execution of this process, since it takes about 2-3 minutes for a protein to predict. After a prediction value is reached, *forcePredictNeed* method is called to determine that whether there is a need for prediction by P2SL or not. This method basically

looks for the case where the prediction is null and it is taken by querying the MEP2SL database. MEP2SL database sometimes does not contain a stored prediction for a protein which belongs to one of the organisms MEP2SL database contains. In this case, user is asked for a confirmation to predict and if confirmed the web service method *GetForcedPrediction* is called. In any stage of the execution, if an error occurs, then the web service response transmits the error details to the plug-in. The prediction request sequence diagram is given in Figure Figure 5.6.

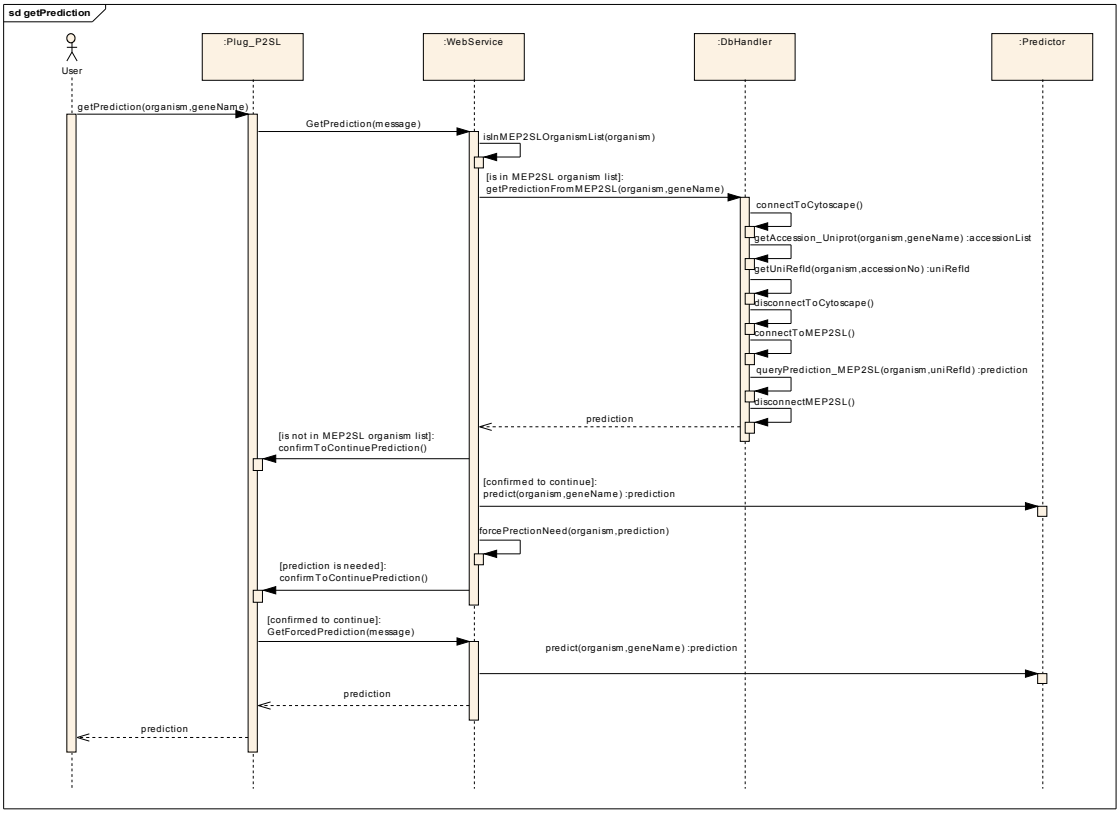


Figure 5.6: Prediction request.

5.4.2 P2SL prediction

Multiuser case is handled in P2SL prediction by the methods *newUserCome* and *newUserGo*. Since P2SL prediction tool needs a separate working environment at each call, with these methods, for each user, a new environment is created. Since P2SL needs sequence information for prediction, at first, *reducedUniprot* database is queried to find the protein sequence

corresponding to the given gene name and organism name. Then P2SL tool is initiated for prediction. The related sequence diagram is given in Figure Figure 5.7.

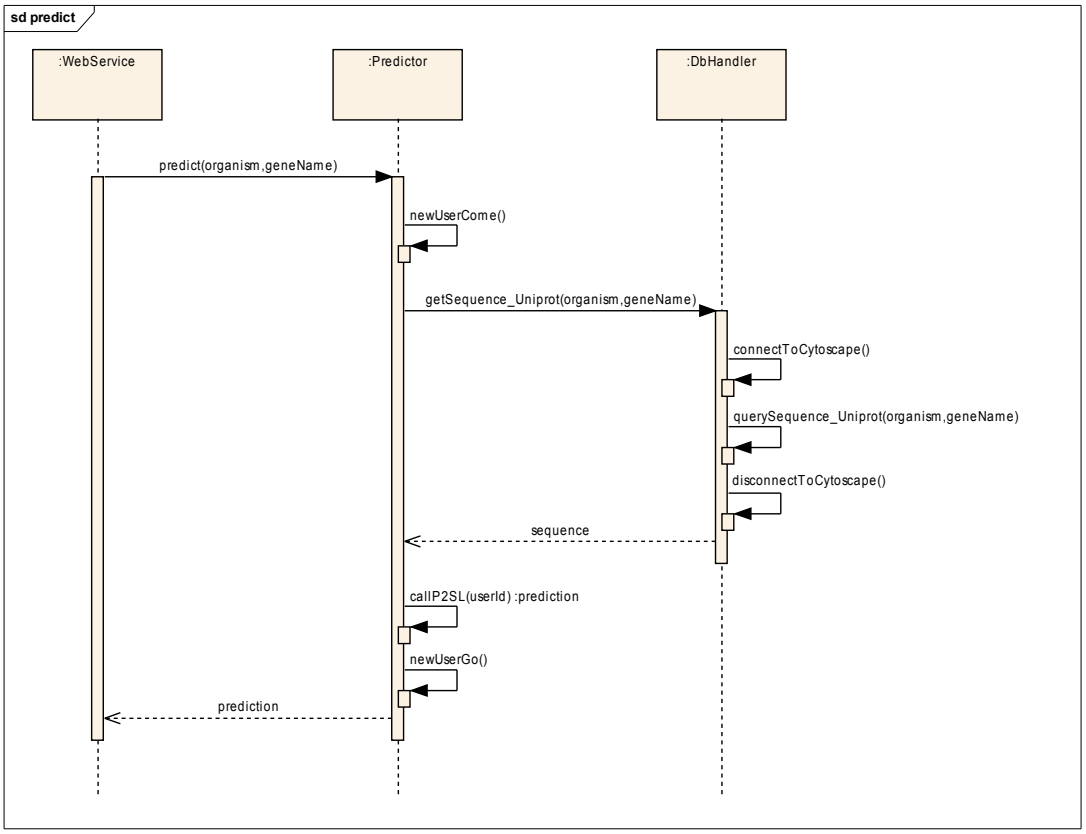


Figure 5.7: P2SL prediction.

5.4.3 Organism list request

User may want to see the organisms related with the gene name of the selected node. After user initiates the process by using the *PlugP2SL* plug-in, the web service method *getMatchingOrganismList* is called with a string argument which is the gene name itself. *reducedUniprot* database is queried and the obtained organism list is packed into a string by separating each organism name with the special character ''. And this string is sent to the plug-in as a response of the web service. The organism list request sequence diagram is given in Figure Figure 5.8.

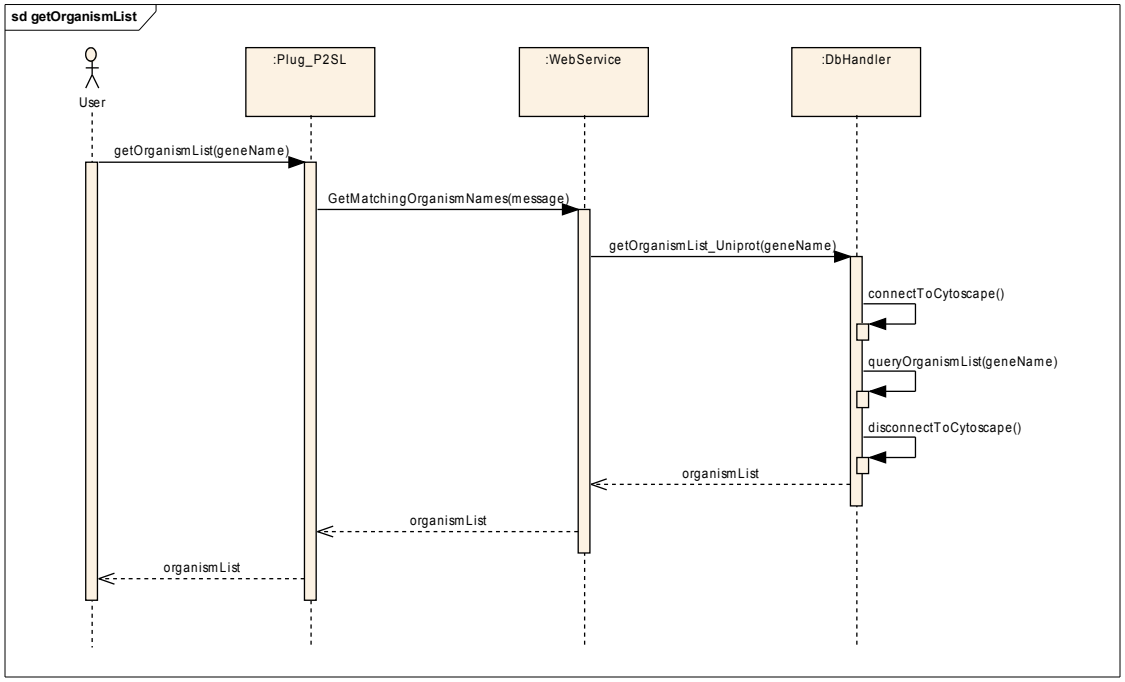


Figure 5.8: Organism list request.

5.4.4 Update internal database

The internal database including the tables *reducedUniprot* and *reducedUniRef* is periodically updated. A timer initiates the update checking process periodically. In this process, UniRef release file is downloaded and compared with the release file that is stored in the last update. If there is a release change until the last update, then update process begins. The initial step is downloading the UniRef100 and UniProtKB/Swiss-Prot databases in xml format. Then, these files are parsed to get the desired fields in the records which are: primary accession number, gene name, organism name and sequence information for UniProtKB/Swiss-Prot database, and Uniref100 id, accession number, organism name for UniRef100 database. After the interested fields are parsed, the internal database is updated with the new records. The related sequence diagram is given in Figure Figure 5.9.

5.5 Class diagrams

Class diagrams are presented in the Appendix A.

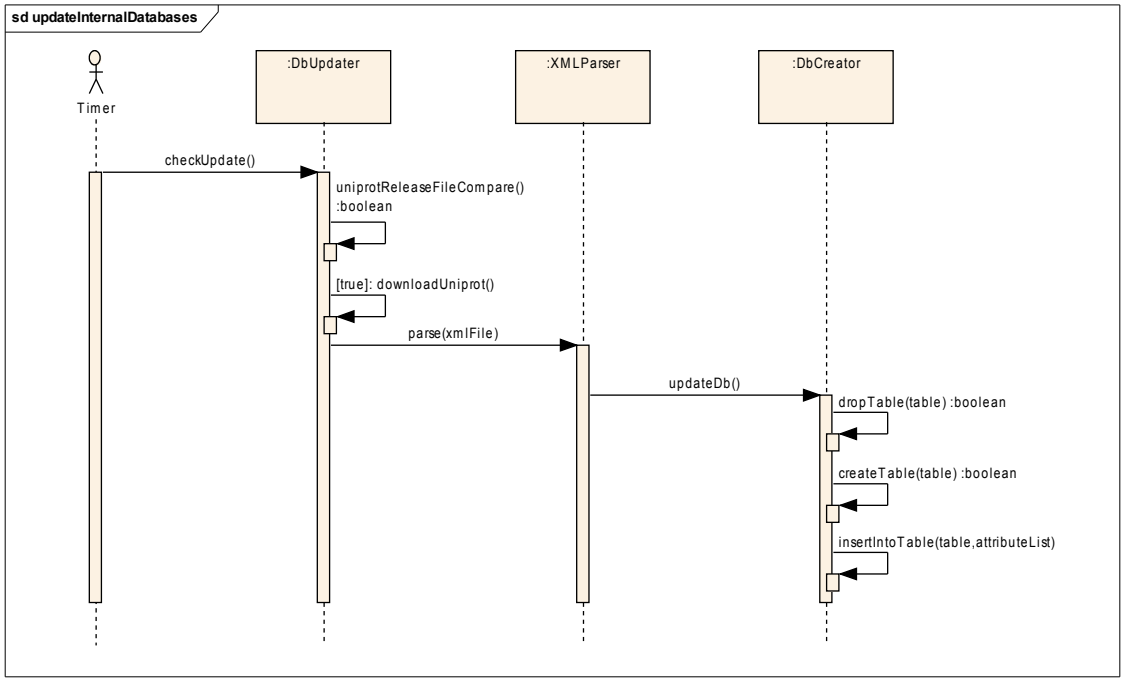


Figure 5.9: Update internal database.

5.6 Implementation

The SOAP server is implemented with Java using Apache AXIS version 1.4. MySQL is used for the database back end and Apache Tomcat is used as the HTTP server daemon. In the construction of the internal database, an XML parser is needed to be employed. SAX API and Piccola XML parser is used for the parser code which is written in Java. *PlugP2SL* is also implemented with Java. For the graphical user interface, Java Swing libraries are used.

5.6.1 SOAP server functions

SOAP server provides 3 functions for remote users which are as follows:

- *GetPrediction* : It takes a string argument which is the concatenation of gene name and organism name separated by a special character '@' and outputs the prediction result.
- *GetMatchingOrganismList* : It takes gene name as the parameter and outputs the related organism names which are concatenated and separated by a special character '@'.

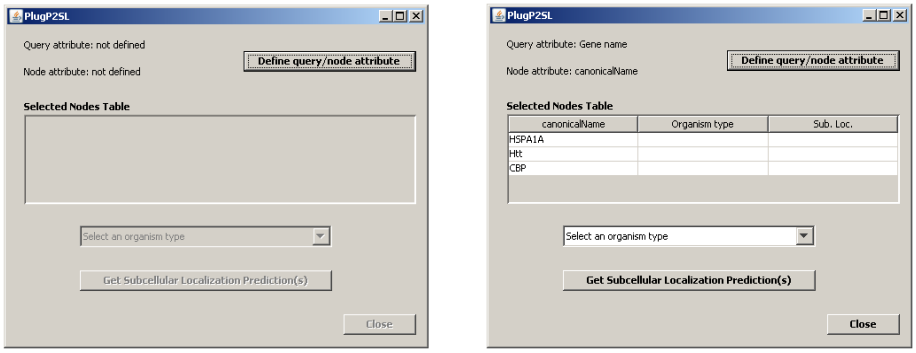
- *GetForcedPrediction* : It is same as *GetPrediction*, but output is forced to be the online prediction result of P2SL, not the MEP2SL query result.

5.6.2 Installation directives of *PlugP2SL*

To install the plug-in download PlugP2SL.rar from
http://www.ceng.metu.edu.tr/ gokcen/PlugP2SL.rar. The compressed archive includes PlugP2SL.jar and its depended libraries. All these files must reside in the "plugin" sub-folder of the Cytoscape program folder. After Cytoscape is started, the plug-in named "PlugP2SL" will appear in the plug-ins list.

5.7 *PlugP2SL* usage

Before employing *PlugP2SL*, user must start Cytoscape, import a network, and import an attribute file which includes the *gene name* attribute. There is no constraint on the name of the attribute, since the attribute which corresponds to the *gene name* attribute can be specified using PlugP2SL plug-in. After selecting one or more nodes, PlugP2SL can be started. The initial PlugP2SL window is given in Figure Figure 5.10(a) where node and query attributes are not defined yet. If there is no defined query and node attributes, plug-in does not let user to do anything before defining the association between query and node attributes. User can define an association using *Attribute Association* form (given in Figure Figure 5.11) that can be reached by *Define query/node attribute* button. Possible query and node attributes are offered in this form. Gene name is the only attribute that can be queried now, but thinking of the future versions, *query attribute* concept is introduced. After an association is defined, the inactive parts of the plug-in form are activated and the selected nodes are listed in the table with columns: defined node attribute, organism type, and subcellular localization prediction. PlugP2SL main window is look like as in Figure Figure 5.10(b) after an association is defined. Plug-in provides users two types of prediction retrieval possibility: individual and batch retrieval. By clicking on a row of the selected nodes table *Individual Retrieval* form is reached. Using this form organism type can be specified. One way of specifying the organism type is to select it from model organisms. The other way is to retrieve the organism list related with the given gene name and then to select the organism type from that list. *Individual Retrieval* form is given in Figure Figure 5.12 in two states showing the two types of organism selection. After organism type is specified, subcellular localization prediction can be fetched. User may prefer to use



(a) Main window with no association definition.

(b) Main window with an association definition.

Figure 5.10: *PlugP2SL* main window.

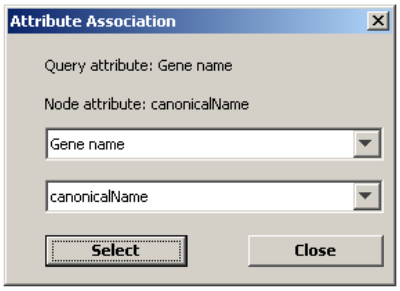
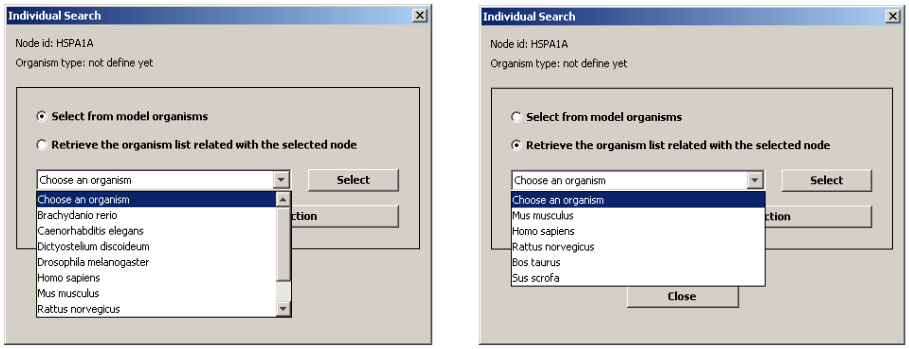


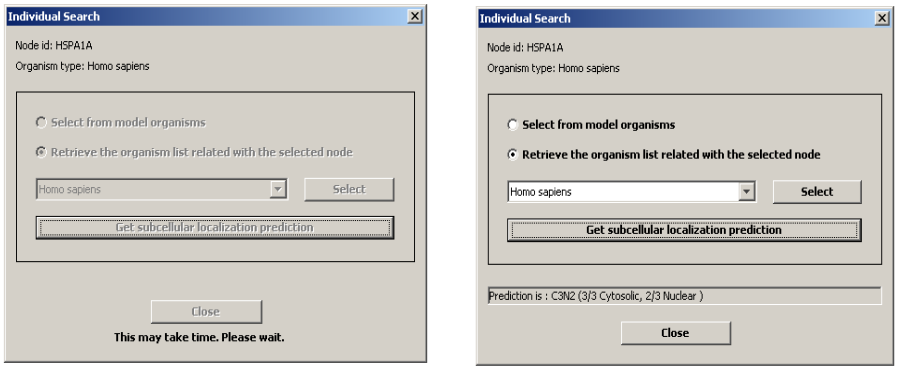
Figure 5.11: Attribute association form.



(a) Model organisms.

(b) Organism list retrieval.

Figure 5.12: Organism selection.

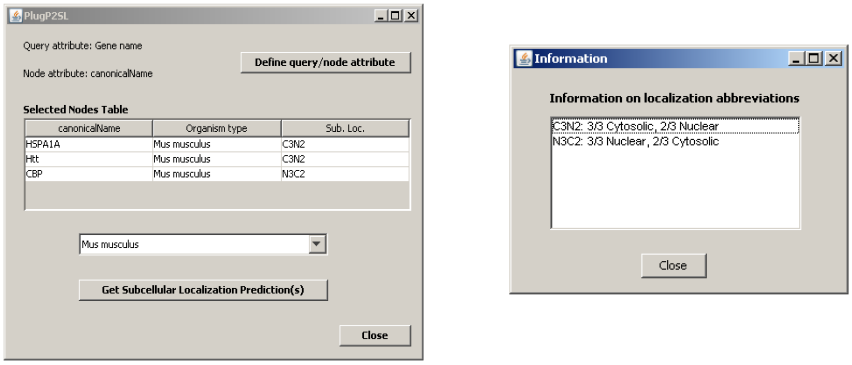


(a) during prediction. (b) after prediction.

Figure 5.13: Individual prediction retrieval.

Individual Retrieval form for only specifying organism types of the nodes and close the form before prediction. This is done when user wants to perform batch prediction with nodes having different organism type specifications. During the prediction fetching process, most features of the *Individual Retrieval* form is deactivated for preventing the user from producing more calls to the system which may confuse the plug-in. *Individual Retrieval* form is given in Figure Figure 5.13 in two phases showing: 'during prediction', and 'after prediction'. Batch prediction retrieval can be done using the main *PlugP2SL* plug-in window. For all the selected nodes listed in the table, an organism type can be specified using the organism type selection combo box. If different organism types are wanted to be specified for one or more nodes, by clicking on the rows of the selected nodes table, *Individual Retrieval* form is reached by which organism type selection can be done. Prediction result is displayed on the last column of the selected nodes table. Since prediction results are abbreviations, *Information* form is displayed after the prediciton for giving information on the abbreviations. Main window of PlugP2SL plug-in after batch prediction and the *Information* form is given in Figure Figure 5.14.

When subcellular localization prediction is not found in the MEP2SL database (i.e. organism type does not belong to the MEP2SL organism list), it must be predicted by P2SL tool on line. Since the process of prediction takes 2-3 minutes, a confirmation is taken from the user for continuing the process. If for a node, the prediction cannot be found in MEP2SL database, but the sequence information can be reached, then in its localization column PREDICT keyword is displayed. If for a node, the prediction cannot be found in MEP2SL database and the sequence information cannot be reached, then in its localization



(a) Main window of *PlugP2SL* plug-in after batch prediction.

(b) *Information* form.

Figure 5.14: Batch prediction retrieval.

column NOT FOUND keyword is displayed. No attribute values set for these kinds of nodes. After subcellular localization prediction retrieval is performed, a number of attributes are defined and added to the selected nodes. Besides the *Localization* attribute, 8 more attributes are added for better analysis of the network. The *Localization* attribute can have 26 different values, i.e. E3, E3N2, E2C2N2, etc. Every value indicates the possible localization predictions, for example, E3N2 means that for the localization prediction of the selected protein, there are 3 votes (which is the maximum vote count a localization compartment can have) to Endoplasmic Reticulum and 2 votes to Nucleus. Each *Localization* attribute contains compartments that take 2 or 3 votes. An additional attribute is defined for each possibility of counts of compartment votes (there are 2 possibilities: 2 or 3). These are: *E2*, *E3*, *N2*, *N3*, *C2*, *C3*, *M2*, *M3*. The ones that constitute the *Localization* attribute take the value 1, and the others take 0 value. The added attributes can be observed from CytoPanel 2 of the Cytoscape window given in Figure Figure 5.15. *PlugP2SL* plug-in is used for annotating biological networks. Cytoscape provides useful functionality for the analysis of the annotated network such as defining a visual style by mapping node colors, shapes, sizes, labels to node attributes, changing the layout of the network by grouping the nodes according to node attributes. An example network on which a number of visual mappings are defined is given in Figure Figure 5.16. In the given network, there are some mapping are defined between node attributes and visual attributes. These are:

- *Localization* attribute and color of the node. Red nodes are C3N2, blue nodes are N3C2, yellow nodes are N3E2, green are E2C2N2, purple are N3M2, etc.

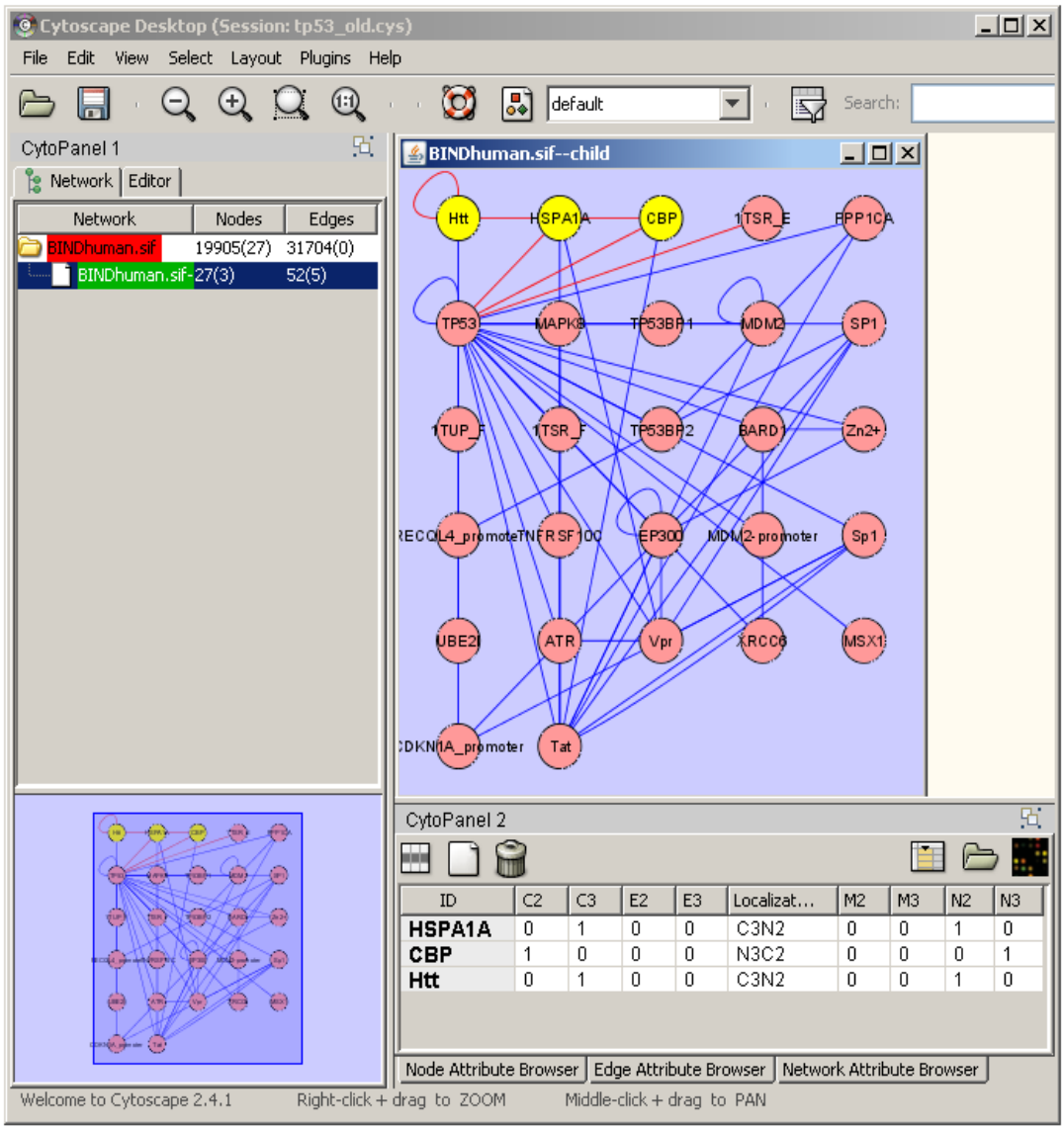


Figure 5.15: Attribute addition.

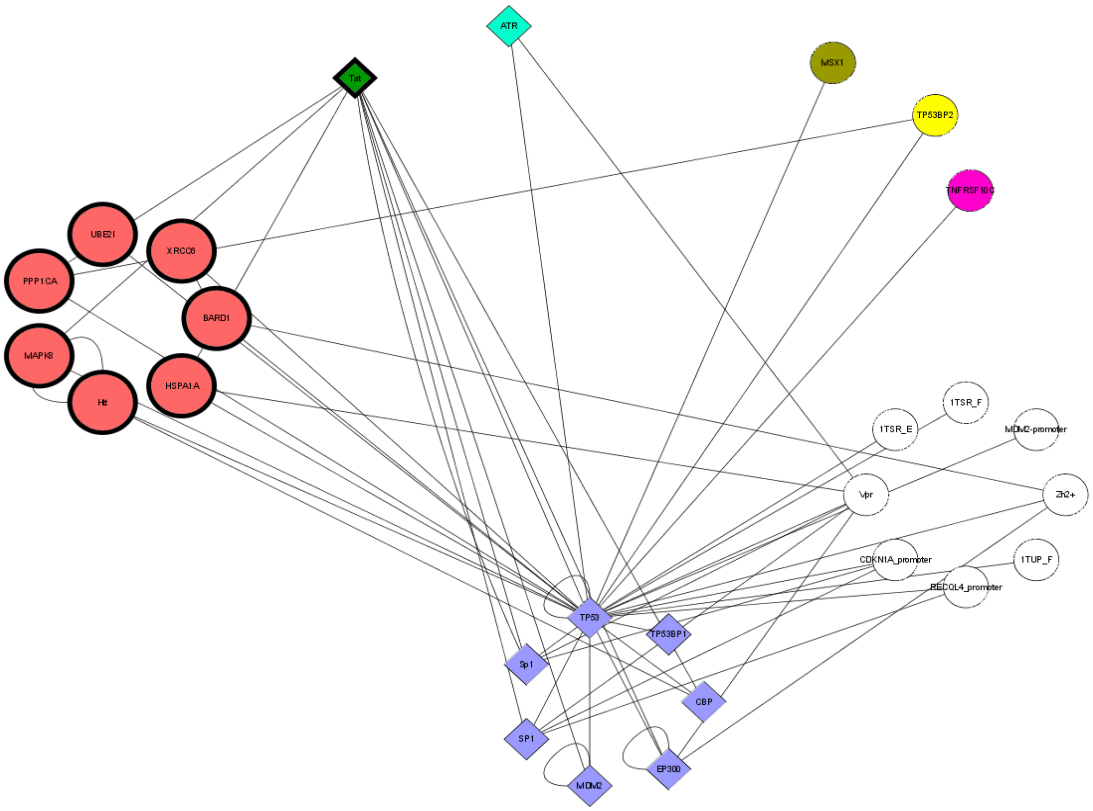


Figure 5.16: An example network analysis.

- *C2* attribute and shape of the node. Circle nodes are 0, square nodes are 1.
- *C3* attribute and size of the node. Smaller nodes are 0, bigger nodes are 1.
- *N2* attribute and line type of the node. Thinner lines are 0, thicker nodes are 1.

CHAPTER 6

CONCLUSION

In the scope of this thesis work, a classification system is built that has two main parts: a subcellular localization prediction tool based on a feature mapping technique to extract biologically meaningful information from protein sequences and a client/server architecture for searching and predicting subcellular localizations. In the first part of the thesis, a feature mapping technique is described which is based on frequent patterns. Frequent patterns in a protein sequence family are identified using a level-wise string-mining algorithm. After the initial feature set is constructed by defining the features as the occurrences of the frequent patterns in a protein sequence, Pearson's chi-squared test is applied on the initial feature set as a feature selection method. A frequency based quantification method is used for determining the numerical values of the features for each sequence. The quantification method is based on the *term-frequency inverted document frequency* (TFIDF) weighting scheme. Support vector machines (SVM) is used for classification. The method is assessed on the subcellular localization prediction problem with 4 compartments: ER targeted (ER), mitochondrial (M), cytosolic (C) and nuclear (N). The dataset is the same used in P2SL. Our method improved the overall accuracy nearly 10 percent which was originally 81.96% by P2SL.

Some alternative feature selection methods are examined including t-test, precision-recall binning, Relief, Adaboost. Pearson's chi-squared test outperforms the other methods in the conducted experiments. Since Adaboost is an embedded feature selection method, instead of SVM, Adaboost is used as the classifier in one of the experiment settings. Being a rule based classification algorithm, Adaboost results are more understandable and readable compared to SVM. We analyzed the Adaboost results for extracting further biological information other than the prediction result. The features that effect the classification result more are extracted for each classification module and the patterns corresponding to these features

are tried to be matched with known biological motifs. The average pattern length changes module to module but it is about 3-4. Since the length of the patterns is short compared to known motifs, we think a number of patterns may cover a known motif in combination. PRINTS database's FPScan tool is used for the analysis. Top ranked patterns (ranked over their effects on classification) are marked on a given sequence and marked fingerprints by FPScan tool on the same sequence are examined for assessing the coverage percent. Although true positive rates are high, which is the fingerprint coverage percent of top ranked patterns, because of the very high false positive rates, the results are not seen meaningful. Due to the extended alphabet definition, the patterns we found are matching with so many subsequences of a protein sequence which leads to the fact that not only the fingerprints marked by FPScan tool, but almost whole sequence is covered by the patterns.

In the second part of the thesis, a client /server architecture is designed and implemented based on Simple Object Access Protocol (SOAP) technology. For the client side of the architecture, a Cytoscape plug-in, PlugP2SL, is developed which helps users to retrieve protein subcellular localization predictions and annotate biological networks visualized in Cytoscape with these information. Most of the time, subcellular localization prediction information does not assist biologists individually. Making analysis of a set of genes/proteins under a system view by integrating information from a number of domains provide more reliable information. Instead of the individual use of subcellular localization information, this plug-in lets biologists to analyze a set of genes/proteins under a system view. Predictions are performed by P2SL tool and MEP2SL database is used for retrieving stored predictions done by P2SL. Predictor unit is designed as a replaceable unit in the system, therefore, instead of P2SL, any predictor can be used.

As future work, frequent pattern search algorithm can be extended to handle gaps, insertions and deletions. In addition to that, some work can be done on the extended alphabet definition to obtain more specific frequent patterns. By means of that, obtained patterns can be validated by using feature rankings acquired from Adaboost results. For the client/server architecture we designed and implemented, it is possible to enhance SOAP server functionalities. For example, besides the 'gene name' identifier, some more identifiers can be added for querying.

REFERENCES

[1] Cytoscape 2.x Plugins. http://www.cytoscape.org/plugins2.php.

[2] Dietterich T. G. Almuallim H. Learning with many irrelevent features. pages 547–552. MIT Press, 1991.

[3] Wu C.H. Barker W.C. Boeckmann B. Ferro S. Gasteiger E. Huang H. Lopez R. Magrane M. Martin M.J. Natale D.A. O'Donovan C. Redaschi N. Apweiler R., Bairoch A. and Yeh L.S. Uniprot: the universal protein knowledgebase. *Nucleic Acids Research*, 32:115–119, 2004.

[4] V. Atalay and R. Cetin-Atalay. Implicit motif distribution based hybrid computational kernel for sequence classification. *Bioinformatics*, 21(8):1429–1436, 2005.

[5] M. Ozturk R. Cetin-Atalay B. Bilen, V. Atalay. http://www.i-cancer.org/mep2sl/.

[6] Greg Barish. Getting started with web services using apache axis. http://www.javaranch.com/newsletter/May2002/axis.html, May 2002.

[7] R. Bellman. *Adaptive Control Processes: A Guided Tour.* Princeton University Press, 1961.

[8] Godelieve L. M Berry-Rogghe. *The computation of collocations and their relevance to lexical studies.* Edinburgh University Press, 1973.

[9] Fabian Birzele and Stefan Kramer. A new representation for protein secondary structure prediction based on frequent patterns. *Bioinformatics*, 2006.

[10] Kenneth W. Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

[11] Hindle D. Church K. W., Hanks P. *Using Statistics in Lexical Analysis.* 1991.

[12] The UniProt Consortium. The universal protein resource (uniprot). *Nucleic Acids Res.*, 35:193–197, 2007.

[13] Béatrice Daille. *Approche mixte pour l'extraction automatique de terminologie: statistiques lexicales et filtres linguistiques.* PhD thesis, Université Paris, 1994.

[14] Christopher Ferris Sharad Garg (eds.) Daniel Austin, Abbie Barbir. http://www.w3.org/TR/wsa-reqs.

[15] Mehran Sahami Daphne Koller. Toward optimal feature selection. In *Proceedings of the 13. ICML*, pages 248–292, 1996.

[16] Morris H. DeGroot and Mark J. Schervish. *Probability and Statistics.* Addison Wesley, 3 edition, 2002.

[17] Sally F. Dennis. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of the Symposium on Statistical Association Methods For Mechanized Documentation*, pages 61–148, 1965.

[18] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.

[19] H. P Edmundson. A correlation coefficient for attributes or events. In *Proceedings of the Symposium on Statistical Association Methods For Mechanized Documentation*, pages 41 – 44, 1965.

[20] Martin Ester and Xiang Zhang. A top-down method for mining most specific frequent patterns in biological sequence data. In *Proceedings of the 4th SIAM International Conference on Data Mining (SDM)*, April 2004.

[21] Stefan Evert. Association measures. http://www.collocations.de/AM/.

[22] Stefan Evert. *The Statistics of Word Cooccurrences: Word Pairs and Collocations.* PhD thesis, University of Stuttgart, 2004.

[23] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[24] Yoav Freund and Robert E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, September 1999.

[25] L. Yu H. Liu. Feature selection for data mining. Technical report, Arizona State University, 2002.

[26] C. W. Hsu and C. J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[27] Ni Huang Hu Chen and Zhirong Sun. Subloc: a server/client suite for protein subcellular location based on soap. *Bioinformatics*, 22(3):376–377, 2005.

[28] Sujun Hua and Zhirong Sun. Support vector machine approach for protein subcellular localization prediction. *Bioinformatics*, 17:721–728, 2001.

[29] Andre Elisseeff Isabelle Guyon. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[30] Masoud Nikravesh Isabelle Guyon, Steve Gunn and ed. Lofti Zadeh. *Feature Extraction: Foundations and Applications*. Physica-Verlag, Springer, 2006.

[31] E. Jerami. *Web services essentials*. OŔeilly Press, 2002.

[32] T. Joachims. *Making large-scale SVM learning practical*. MIT Press, 1999.

[33] Julian Tonti Filippini Ian Small Joshua L.Heazlewood, Robert E. Verboom and A. Harvey Millar. Suba: the arabidopsis subcellular database. *Nucleic Acids Research*, 35:213–218, 2007.

[34] Walter A. Kosters Kai Ye and Adriaan P. IJzerman. An efficient, versatile and scalable pattern growth approach to mine frequent patterns in unaligned protein sequences. *Bioinformatics*, 23:687–693, 2007.

[35] K. Kira and L. A. Rendell. The feature selection problem: Traditional methods and a new algorithm. In San Jose, editor, *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 129–134, 1992.

[36] I. Kononenko. Estimating attributes: Analysis and extensions of relief. In L. De Raedt and F. Bergadano, editors, *Proceedings of the European Conference on Machine Learning*, pages 171–182. Springer Verlag, 1994.

[37] Huan Liu Lei Yu. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, 5:1205–1224, 2004.

[38] Binns D. Fleischmann W. Lopez R. Apweiler R Leinonen R., Diez F.G. Uniprot archive. *Bioinformatics*, 20:3236–3237, 2004.

[39] Douglas Liddell. Practical tests of 2 x 2 contingency tables. *The Statistician*, 25(4):295–304, 1976.

[40] H Mannila and H Toivonen. Level-wise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 3(1):241–258, 1997.

[41] The UniProtKB User Manual. http://www.expasy.org/sprot/userman.html.

[42] Rajesh Nair and Burkhard Rost. Locnet and loctarget: sub-cellular localization for structural genomics targets. *Nucleic Acids Research*, 32:517–521, 2004.

[43] Horton P. Nakai K. Psort: a program for detecting sorting signals in proteins and predicting their subcellular localization. *Trends Biochem Sci.*, 24(1):34–36, January 1999.

[44] S. Brunak O. Emanuelsson, H. Nielsen and G. von Heijne. Predicting subcellular localization of proteins based on their n-terminal amino acid sequence. *Journal of Molecular Biology*, 300:1005–1016, 2000.

[45] Melissa Cline Micheline Fromont Racine Alain Jacquier Benno Schwikowski Olivier Garcia, Cosmin Saveanu and Tero Aittokallio. Golorize: a cytoscape plug-in for network visualization with gene ontology-based layout and coloring. *Bioinformatics*, 23(3):394–396, 2006.

[46] Randy Goebel Gregory Taylor Osmar R. Zaiane, Yang Wang. Frequent subsequence-based protein localization. pages 35–47. Springer Verlag, 2007.

[47] Owen Ozier Nitin S. Baliga Jonathan T. Wang Daniel Ramage Nada Amin Benno Schwikowski Paul Shannon, Andrew Markiel and Trey Ideker. Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Res.*, 13(11):2498–2504, 2003.

[48] Ted Pedersen and Rebecca Bruce. What to infer from a description. Technical Report 96-CSE-04, Southern Methodist University, Dallas, 1996.

[49] Uwe Quasthoff and Christian Wolff. The poisson collocation measure and its application. 2002.

[50] Marko Robnik-Sikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine Learning Journal*, 53:23–69, 2003.

[51] George H. John Ron Kohavi. Wrappers for feature subset selection. *Artificial Intelligence*, 97:273–374, 1997.

[52] Ke Wang Martin Ester Jennifer L. Gardy Fiona S. L. Brinkman Rong She, Fei Chen. Frequent-subsequence-based prediction of outer membrane proteins. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining table of contents*, pages 436–445, 2003.

[53] G Salton and C Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 513(24), 1988.

[54] Gursoy-Yuzugullu O. Cetin-Atalay R. Sarac, O.S. and V. Atalay. Protein function annotation by subsequence based feature map. 2007.

[55] Unified Modeling Language Web Site. http://www.uml.org.

[56] Frank Smadja. Retrieving collocations from text: Xtract. *Computational Linguistics*, 19(1):143–177, 1993.

[57] Doug; Kulchenko Pavel Snell, James; Tidwell. *Programming WebServices with SOAP*. OŔeilly Press, 1 edition, 2002.

[58] Karel Heymans Steven Maere and Martin Kuiper. Bingo: a cytoscape plugin to assess overrepresentation of gene ontology categories in biological networks. *Bioinformatics*, 21(16):3448–3449, 2005.

[59] Contingency table definition. http://www.answers.com/topic/contingency-table?cat=biz-fin.

[60] Xinglai Ji Tao Guo, Sujun Hua and Zhirong Sun. Dbsubloc: database of protein subcellular localization. *Nucleic Acids Research*, 32:122–124, 2004.

[61] V Vapnik. *Statistical learning theory*. Wiley-Interscience, 1998.

[62] V. Vasudevan. A web services primer. http://webservices.xml.com/pub/a/ws/2001/04/04/webservices 2001.

[63] Jean-Philippe Vert. Introduction to support vector machines and applications to computational biology. *DRAFT*, 2001.

[64] Alexander Vezhnevets. http://research.graphicon.ru/machine-learning/gml-adaboost-matlab-toolbox.html.

[65] World Wide Web Consortium (W3C). Soap version 1.2: Messaging framework. http://www.w3.org/TR/SOAP, 2003.

[66] Wei Wang and Jiong Yang. Mining high dimensional data, mining sequential patterns from large data sets. *The Kluwer International Series on Advances in Database Systems*, 28, 2005.

[67] W.R.Taylor. The classification of amino acid conservation. *J Theor Biol.*, 119(2):205–218, March 1986.

[68] F. Yates. Tests of significance for 2 x 2 contingency tables. *Journal of the Royal Statistical Society, Series A*, 147(3):426–463, 1984.

[69] Mamun Zaman. An introduction to web services. http://www.devarticles.com/c/a/Web-Services/An-Introduction-to-Web-Services, May 2007.

Appendix A

CLASS DIAGRAMS

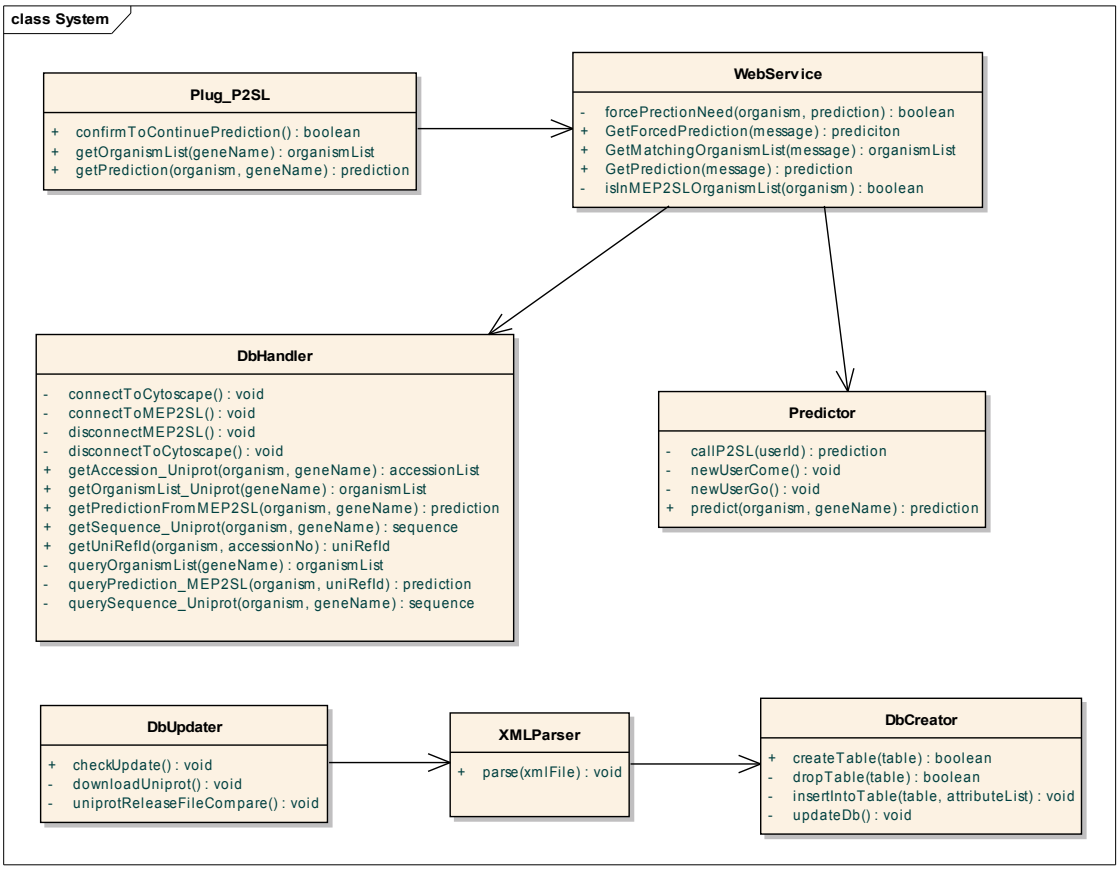


Figure A.1: Class Diagram Relations.

A.1 *PlugP2SL*

This class works on the client side. It basically takes user requests and preferences, calls the provided web services, and outputs the web service responses to the user.

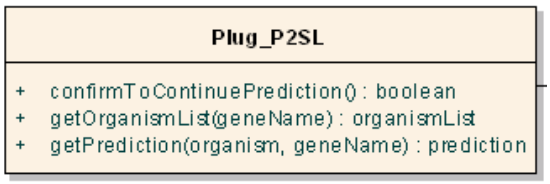


Figure A.2: *PlugP2SL* Class.

A.1.1 Methods

boolean confirmToContinuePrediction()

This method asks user for a confirmation to continue the prediction process by P2SL. If user confirms to continue, then the web service method *GetForcedPrediction* is called.

organismList getOrganismList(geneName)

By this method, the web service method *GetMatchingOrganismList* is called with a gene name parameter. The called web service method returns the organism list as output. The list is output to the user.

prediction getPrediction(organism, geneName)

By this method, the web service method *GetPrediction* is called with gene name and organism name parameters. The called web service method returns the prediction as output. The prediction is output to the user.

A.2 WebService

This class works on the server side. It provides a number of web services for remote usage, and fulfils the main functionality of the system by using *DbHandler* and *Predictor* classes.

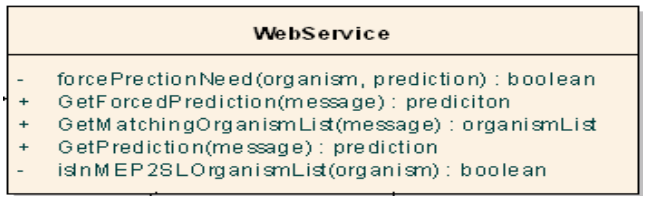


Figure A.3: WebService Class.

A.2.1 Methods

boolean forcePredictionNeed(organism, prediction)

This method examines the prediction and the organism name. If the prediciton is null and the organism name belongs to the organisms MEP2SL database contains, then there is a possibility to find the predicition by using P2SL tool. But before continuing the process, user is asked for confirmation.

prediction GetForcedPrediction(message)

This method directly initiates the prediction process. *predict* method which is provided by the *Predictor* class is called. The output of this method, which is the prediction, is output to web service caller.

organismList GetMatchingOrganismList(message)

This method retrieves the related organism list with the given gene name. *getOrganismListUniprot* method which is provided by the *DbHandler* class is called. The result is output to web service caller.

prediction GetPrediction(message)

This is the general method for retrieving prediction. At first, the organism name is examined by *isInMEP2SLOrganismList* method. If the given organism name belongs to the organisms MEP2SL database contains, then *getPredictionFromMEP2SL* method which is provided by the *DbHandler* class is called, else *predict* method of the *Predictor* class is called. The result is output to web service caller.

boolean isInMEP2SLOrganismList(organism)

If the given organism name belongs to the organisms MEP2SL database contains,this method returns true, else returns false.

A.3 DbHandler

This class works on the server side. It provides access to the internal database including the tables *reducedUniprot* and *reducedUniRef* and to the external database *MEP2SL*.

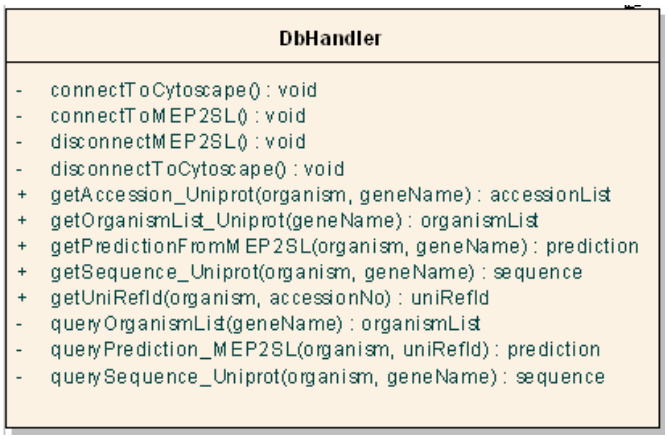


Figure A.4: DbHandler Class.

A.3.1 Methods

void connectToCytoscape()

This method connects to the internal database.

void disconnectToCytoscape()

This method disconnects from the internal database.

void connectToMEP2SL()

This method connects to the external database *MEP2SL*.

void disconnectMEP2SL()

This method disconnects from the internal database.

organismList getOrganismListUniprot(geneName)

This method is called by the *WebService* class when a request comes on retrieving the organism list related with the given gene name. After the connection is established to the internal database, *reducedUniprot* table is queried by the *queryOrganismList* method and the connection is closed. The retrieved list is then output to the caller method.

prediction getPredictionFromMEP2SL(organism, geneName)

This method is called by the *WebService* class when *GetPrediction* method is called and *isInMEP2SLOrganismList* method gives 'true' as an output. It initiates the process of retrieving the prediction from MEP2SL database. Internal database is used for accessing the uniRef id of the related record and then by using this id MEP2SL database is queried for the prediciton. The retrieved prediction is then output to the caller method.

sequence getSequenceUniprot(organism, geneName)

This method is called by the *Predictor* class when *predict* method is called. After the connection is established to the internal database,*reducedUniprot* table is queried by the *querySequenceUniprot* method and the connection is closed. The retrieved sequence information is then output to the caller method.

accessionList getAccessionUniprot(organism, geneName)

This method queries the *reducedUniprot* table for accessing the accession numbers related with the given protein. These numbers are then used for finding the uniref id of the related record.

uniRefId getUniRefId(organism, accessionNo)

This method queries the *reducedUniRef* table for accessing the uniRef id of a protein with the given accession number and organism name.

prediction queryPredictionMEP2SL(organism, uniRefId)

This method queries the *MEP2SL* database for accessing the subcellular localization prediction of a protein with the given uniRef id and organism name.

organismList queryOrganismList(geneName)

This method queries the *reducedUniprot* table for accessing the organism names related with the given protein.

sequence querySequenceUniprot(organism, geneName)

This method queries the *reducedUniprot* table for accessing the sequence information related with the given protein.

A.4 Predictor

This class works on the server side. It provides access to the P2SL predictor, handles multiuser usage of the predictor and outputs the retrieved predicition to the caller method.

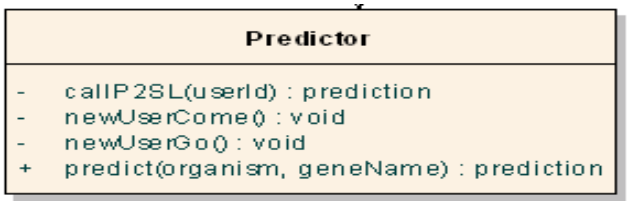


Figure A.5: Predictor Class.

A.4.1 Methods

prediction predict(organism, geneName)

This is the main method of this class. After accessing the sequence information of the given protein by using the *getSequenceUniprot* method of the *DbHandler* class, *callP2SL* is called for initiating the P2SL prediction process. *newUserCome* and *newUserGo* methods are called at the beginning and at the end of the method respectively for handling the multiuser usage of the P2SL tool.

prediction callP2SL(userId)

This method initiates the P2SL prediction process. Outputs the result given by the P2SL tool.

void newUserCome()

This method creates a seperate working environment for each user, before calling the P2SL tool.

void newUserGo()

This method destroys the created working environment after calling the P2SL tool.

A.5 DbUpdater

Figure A.6: DbUpdater Class.

This class works on the server side. It handles the regular update of the internal database tables.

A.5.1 Methods

void checkUpdate()

This method checks whether there is a need for update by calling the *uniprotReleaseFile-Compare* method. If a new release is announced, then it initializes the update process by calling the *downloadUniprot* method and *parse* method of the *XMLParser* class.

boolean uniprotReleaseFileCompare()

This method downloads the release file of Uniprot and compares it with the file stored in the system since the last update. If version number is changed, then it returns true, else returns false.

void downloadUniprot()

This method downloads the UniRef100 and UniProtKB/Swiss-Prot databases in xml format.

A.6 XMLParser

This class works on the server side. It parses the given XML files.

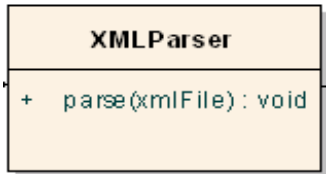


Figure A.7: XMLParser Class.

A.6.1 Methods

void parse(xmlFile)

This method basically parses the given XML file.

A.7 DbCreator

This class works on the server side. It handles the reconstruction of the internal database tables in the update process.

A.7.1 Methods

boolean createTable(table)

This method is used for the creation of the internal database tables.

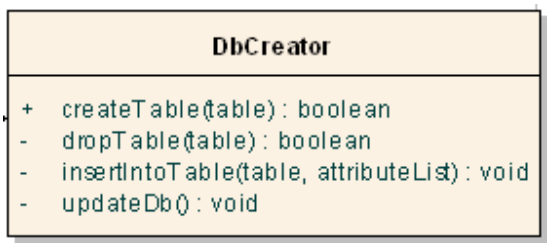


Figure A.8: DbCreator Class.

boolean dropTable(table)

This method is used for the deletion of the internal database tables.

void insertIntoTable(table, attributeList)

This method is used for inserting the records with the given fields to the specified tables.

void updateDb()

This method initiates the reconstruction of the internal database tables.