University of Alberta

EPPDB: A DATABASE FOR PROTEOMIC ANALYSIS OF EXTRACYTOSOLIC PLANT PROTEINS

by

Yang Wang 

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Fall 2004

Your file Votre référence
ISBN: 0-612-95876-0
Our file Notre référence
ISBN: 0-612-95876-0

To my family, for their endless love and support.

Acknowledgements

I have been priviledged to have had the opportunities to work with many brilliant people during my thesis research. Without their help and guidance, this thesis work would be impossible in many ways.

First and foremost, I would like to express my deepest gratitude to my supervisors, Dr. Osmar R. Zaïane and Dr. Randy Goebel. I have benefited a lot from their insightful guidance, inspiring support and invaluable knowledge. I am also very grateful for their financial support and many reference letters.

I would like to thank Dr. Michael Deyholos for being my external examiner and providing many helpful comments on my thesis.

This work could not have been done without the generous assistance of many biologists that I have been working with, including Jennafer L. Southron, Urmila Basu, Randy M. Whittal, Julie L. Stephens and Gregory Taylor. They have taught me a lot about the biological background of this thesis work.

Thanks also go to Rob Lake, Maria-Luiza Antonie and Zhiyong Lu. Rob helped me on many technical issues during the early stage of this work. Luiza shared her work with me and provided me a lot of advices. Zhiyong helped me with the plant dataset used in this work.

Last, but definitely not the least, I would like to thank my parents, my sisters and my sister-in-laws for many years of unconditional love, support and understanding. The pursuit of graduate study is partially a result of their encouragement for advanced education since my childhood. They are what makes all the efforts worthwhile. It is on their behalf I dedicate this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Background

A proteome represents the proteins that are expressed in a specific biological unit at a particular time and under a particular set of conditions. Proteomics utilizes a diverse set of tools to display, identify, and investigate the proteins in a proteome. The results of proteomic studies are commonly displayed in on-line databases. In particular, experimental data available from two-dimensional polyacrylamide gel electrophoresis (2-D PAGE) are displayed in 2-D PAGE databases (or 2-DE databases) [5]. Links to many 2-D PAGE database servers and 2-D PAGE related servers and services can be found at WORLD-2DPAGE[1] and efforts have been made to establish a set of federated databases [5] that are maintained independently, but are linked together through the World Wide Web (WWW).

This dissertation describes the design and implementation of *EPPdb (Extracytosolic Plant Protein Database)*, an on-line 2-D PAGE database that is built to provide the plant biology community with relevant information about extracytosolic plant proteins. Extracytosolic plant proteins are involved in numerous processes including nutrient acquisition, communication with other soil organisms, protection from pathogens, and resistance to disease and toxic metals. Insofar as these proteins are strategically positioned to play a role in resistance to environmental stress, biologists are using proteomic tools, such as two-dimensional polyacrylamide gel electrophoresis (2-D PAGE), liquid

[1] http://ca.expasy.org/ch2d/2d-index.html

chromatography-tandem mass spectrometry (LC-MS/MS), *de novo* sequencing, and bioinformatics, to analyze extracellular proteins. These proteins are collected from *Brassica napus* (canola) plants grown hydroponically in a sterile environment[4, 8, 81]. The goal of this database is to allow biologists retrieve and submit information about extracytosolic plant proteins, and to perform advanced data analysis and data mining tasks using tools submitted to the database by researchers.

1.2 Contributions

The contributions of this dissertation include:

- From the biology side, EPPdb is the first database built solely for extracytosolic plant proteins. The information will be available for use in plant physiology and plant breeding programs throughout the world to improve crop growth. It will open up new opportunities for discovery of novel genes and promoters.
- EPPdb uses Web Services to facilitate information sharing among applications, which is quite novel in biological database management community.
- EPPdb introduces the concept of allowing users to submit tools to facilitate more sophisticated data analysis and data mining tasks.
- As a proof of the concept of data analysis and data mining tools, a tool for identifying extracellular proteins from primary sequences is built by using data mining techniques.

1.3 Thesis Organization

The remainder of this dissertation is organized as follows. Chapter 2 and Chapter 3 comprise the first part of this dissertation, which focuses on the 2D-PAGE database. Chapter 2 introduces some related work on protein databases. Chapter 3 describes the design and implementation of various functionalities of

the database. The second part of this dissertation concentrates on a database analysis tool that we build for EPPdb, which is to identify extracellular proteins from protein primary sequences. This part includes Chapter 4, Chapter 5 and Chapter 6. Chapter 4 introduces biological background and existing methods that are related to the problem of extracellular protein prediction. This chapter also introduces the problem of mining sequential data, which is related to the methods we use. Chapter 5 describes the approaches we use for predicting extracellular proteins, including feature extraction, sequence modeling, and learning algorithms. Chapter 6 presents the experimental results on real-world datasets. The last chapter (Chapter 7) concludes the dissertation and points to some future work. Appendix A gives a short glossary of common terms. Appendix B shows a sample code for accessing our database through Web services.

Chapter 2

Work Related to Protein Databases

2.1 Introduction

With the availability of over 165 completed genome sequences from both eukaryotic and prokaryotic organisms, a huge volume of data are generated from the large-scale analysis of these proteins. This comes from both the information provided by the genome projects and the newly developed technologies in protein science. Nowadays, it is possible to quickly identify large number of proteins, to map their interactions, to determine their locations within the cell and to analyze their biological activities. Protein sequence databases play an important role as a repository for storing the accumulated data and making them accessible to the scientific community [7].

This chapter introduces the work related to the protein database. According to the level of additional information to the sequence records they contain, protein sequence databases can be categorized as sequence repositories and curated databases. Section 2.2 and Section 2.3 introduce these two type of protein databases, respectively. Section 2.4 introduces the two-dimensional electrophoresis (2-DE) database, which is the type of databases EPPdb falls into [7]. Section 2.5 summarizes this chapter.

2.2 Sequence Repositories

Several protein sequence databases act as repositories of protein sequences, in the sense that they contain little or no additional information other than protein sequences. Usually they make no effort to provide a non-redundant collection of sequences to users.

An example of sequence repositories is GenBank Gene Products (GenPept) database. The entries in the database are derived from translations of the sequences contained in the nucleotide database maintained by the DNA Database Bank of Japan (DDBJ), the European Molecular Biology Laboratory (EMBL) Nucleotide Sequence Database and GenBank. The entries in the database contain minimal annotation extracted from the corresponding nucleotide entries. These databases lack additional annotation and do not contain proteins derived from amino acid sequencing. Also, each protein in the database can be represented by different entries, so the database is redundant.

2.3 Curated Databases

Compared with sequence repositories, the curated databases enrich the sequence data in the databases by adding additional information. The additional information is usually validated by expert biologists before they are added into the databases. Also effort is made to remove redundancy of the databases by compiling all reports for a given protein sequence into a single entry.

SWISS-PROT[1] is the most well-known universal curated protein sequence database. It contains 152040 sequence entries as of May 2004 (Release 43.4). In SWISS-PROT, all the reports for a given protein are merged into a single entry, so the database is non-redundant.

The core data, which is required for every SWISS-PROT entry, consists of the amino acid sequence, the protein name (description), taxonomy data and citation information. Additional information on proteins is available, the

[1] http://www.ebi.ac.uk/swissprot/index.html

entries contain detailed annotation on items such as the functions of the protein, subcellular locations, secondary structure, tertiary structure, quaternary structure, etc. The annotation added is stored mainly in the description (DE) lines, the gene (GN) lines, the comment (CC) lines, the feature table (FT) lines, and the keyword (KW) lines [9].

2.4 2-DE Databases

A two-dimensional electrophoresis (2-DE) database is a specific protein database that contains experimental data available from two-dimensional polyacrylamide gel electrophoresis (2-D PAGE). A 2-DE database usually contains textual descriptions of the proteins identified and various 2-D PAGE images showing the protein locations.

The most famous 2-DE database is SWISS-2DPAGE[2]. Each SWISS-2DPAGE entry corresponds to one protein and contains several lines to represent the textual descriptions of the protein, including mapping procedures, physiological and pathological information, experimental data (isoelectric point, molecular weight, amino acid composition, peptide masses) and bibliographical references. Each line begins with a two-character line code, which indicates the type of data contained in the line. Table 2.1 shows the lines codes and line types[3] used in SWISS-2DPAGE.

In addition to this textual data, SWISS-2DPAGE provides several 2-D PAGE and SDS-PAGE images showing the experimentally determined location of the protein, as well as a theoretical region computed from the protein sequence, indicating where the protein might be found in the gel. Cross-references are provided to Medline and other federated 2-DE databases (YEPD, ECO2DBASE, HSC-2DPAGE, PHCI-2DPAGE, PMMA-2DPAGE, Siena-2DPAGE) and to SWISS-PROT, which provides many links to other molecular databases (EMBL, Genbank, PROSITE, OMIM, etc).

SWISS-2DPAGE can be browsed through the ExPASy World Wide Web

[2]http://au.expasy.org/ch2d
[3]http://tw.expasy.org/ch2d/manch2d.html

Line code	Line type	Occurrence in an entry
ID	Identification	Once; starts the entry
AC	Accession number(s)	One or more
DT	Date	Two times
DE	Description	One or more
GN	Gene name(s)	Optional
OS	Organism species	One or more
OC	Organism classification	One or more
OX	Taxonomy cross-reference(s)	Once
MT	Masters	One or more
IM	Images	One or more
RN	Reference number	One or more
RP	Reference position	One or more
RX	Reference cross-reference(s)	Optional
RA	Reference authors	One or more
RT	Reference title	Optional
RL	Reference location	One or more
CC	Comments or notes	One or more
2D	2-D PAGE specific data	several
1D	SDS-PAGE specific data	several
DR	Database cross-reference	Optional
//	Termination line	Once; ends the entry

Table 2.1: Line codes and line types in SWISS-2DPAGE

molecular biology server. The SWISS-2DPAGE top page provides several text searches, and displays results with links to other databases. SWISS-2DPAGE also allows users to select a 2-D PAGE map that will be displayed, then the user can click a spot in the map and obtain the information on the corresponding protein.

2.5 Summary

This chapter has presented some work related to protein database. In particular, we introduced 2-DE database, a concept upon which EPPdb will be built. The next chapter will describe in detail the design and implementation of the various functionalities of EPPdb.

Chapter 3

Extracytosolic Plant Protein Database

3.1 Introduction

EPPdb contains the results from proteomic studies on proteins collected from *Brassica napus (canola)* plants, including the 2D maps showing the protein locations, and descriptions of the identified proteins. Cross references are provided to SWISS-PROT/TrEMBL [9], which is the largest annotated protein database in the world. In order to get the data for our database, the proteins in the cells or tissues to be studied are solubilized by biologists. The DNA and other contaminants are removed. The proteins are separated by their charges using isoelectric focusing, which is an electrophoresis between a cathode and anode with the cathode at a higher pH than the anode. Because the amino acids in proteins have amphoteric properties, they will be separated by migrating toward different pH values. After that, the separated proteins are analyzed by LC/MS (Liquid Chromatography/Mass Spectrometry). The resulting LC/MS data are submitted for database searching. After *de novo* sequencing the proteins, they are submitted to the MS-Blast and MS-Homology to find homologous proteins in other plant species [4, 8].

What distinguishes our database from other 2-D PAGE databases is that it not only provides a Web interface for querying using a client browser, but also provides Web services that allow other applications to make function calls over HTTP and use XML as a message transfer format to be consumed by

the clients. To the best of our knowledge, no other major biological databases support an XML interface.

Our database also aims to provide tools that facilitate more sophisticated data analysis and data mining tasks. To achieve this goal, our database is built as a framework that allows users to submit not only queries, but also data and tools for experimenting with various data analysis and data mining tasks. For example, one tool that we have developed in this dissertation is for predicting extracellular proteins from amino acid sequences.

The sections that follow elaborate on the design and implementation of our database. Section 3.2 introduces the overall architecture of our database. Section 3.3 discusses the implementation of search functions. Section 3.4 introduces the Web services implemented as part of our database. Section 3.5 describes the tools added to our database for experimenting with various data analysis and data mining tasks.

3.2 System Architecture

The overall architecture of the our database is shown in Figure 3.1. The protein database contains all the protein entries. The format of protein entries is similar to that in SWISS-PROT/TrEMBL [9] and SWISS-2DPAGE [30]. Each entry is composed of defined lines, used to record various kinds of data. Each line begins with a two character line code, which indicates the type of data contained in the line[1]. Table 3.1 shows the line codes and line types used in EPPdb. An example of a protein entry is shown in Figure 3.2. Several lines are specific to our database: (i) the DB line lists an accession number specific to our database. Each entry in our database has a unique DB line; (ii) if applicable, the IS line lists the isozyme(s) (identified by their DB accession numbers) of a protein. In addition, the 2-D map associated with a protein entry displays the experimental location of the protein on the chosen map (Figure 3.3).

The protein entries in the database can be queried in two ways. Firstly, a

[1]Uniprot/SWISS-PROT User Manual, http://ca.expasy.org/sprot/userman.html

Line code	Line type	Occurrence in an entry
ID	Identification in SWISS-PROT	Once; starts the entry
AC	Accession number(s)	One or more
DT	Date	Two times
DE	Description	One or more
GN	Gene name(s)	Optional
OS	Organism species	One or more
OC	Organism classification	One or more
OX	Taxonomy cross-reference(s)	Once
MT	Masters	One or more
IM	Images	One or more
RN	Reference number	One or more
RP	Reference position	One or more
RA	Reference authors	One or more
RT	Reference title	Optional
RL	Reference location	One or more
CC	Comments or notes	One or more
2D	2-D PAGE specific data	several
DR	Database cross-reference	Optional
DB	Identification in EPPdb	Once
IS	Other protein spots matching the top hit	Optional
SQ	Protein sequence	Once
KW	Keywords	Optional
AL	Alignment	Optional
OP	Other sequenced peptides	Optional
//	Termination line	Once; ends the entry

Table 3.1: Line codes and line types in EPPdb

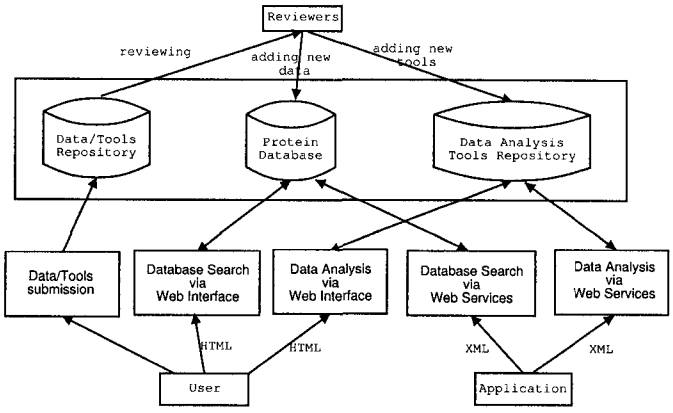


Figure 3.1: Overall architecture of EPPdb

```
ID Q9M8Y9; PRELIMINARY; 2DG.
AC Q9M8Y9;
DT 30-MAY-2003 (Rel.  01, Created)
DT 30-MAY-2003 (Rel.  01, Last update)
DE Putative trypsin inhibitor.
GN T6K12.5.
OS Arabidopsis thaliana (Mouse-ear cress).
OC Eukaryota; Viridiplantae; Streptophyta; Embryophyta; Tracheophyta;
OC Spermatophyta; Magnoliophyta; eudicotyledons; core eudicots; Rosidae;
OC eurosids II; Brassicales; Brassicaceae; Arabidopsis.
OX NCBI_TaxID=3702;
MT <i>Brassica napus</i> Extracellular Proteome.
IM <i>Brassica napus</i> Extracellular Proteome.
RN [1]
RP MAPPING ON GEL.
RA Basu U., et al.;
RT ''Using proteomics to establish an Extracytosolic Plant Proteins Database'';
RL Unpublished observations (MAY-2003).
CC -!- SUBCELLULAR LOCATION: Secretory pathway signal peptide (predicted by TargetP; RC 1).
CC -!- PTM: SignalP predicts most likely cleavage site to be between pos.  21 and 22 (TSG-VV).
CC -!- MISCELLANEOUS: Predicted pI 5.90.
2D -!- MASTER: BRASSICA_NAPUS_EXTRACELLULAR_PROTEOME;
2D -!- PI/MW: SPOT 00001=5.0/23000;
2D -!- MAPPING: SPOT 00001:  LC-MS/MS.
2D -!- PEPTIDE SEQUENCES: SPOT 00001:  FANPSKCGESGVWR; VANGEVVLNGVESR;
2D CPHQPVMF; SCKGSLSWETGAAEGN; LLPSSTV.
DR TrEMBL; Q9M8Y9; Q9M8Y9.
DB 00001.
IS 00002;
SQ SEQUENCE 202 AA; 22914 MW; 485A2C8472CD3792 CRC64;
KW
//
```

Figure 3.2: An example of a database entry

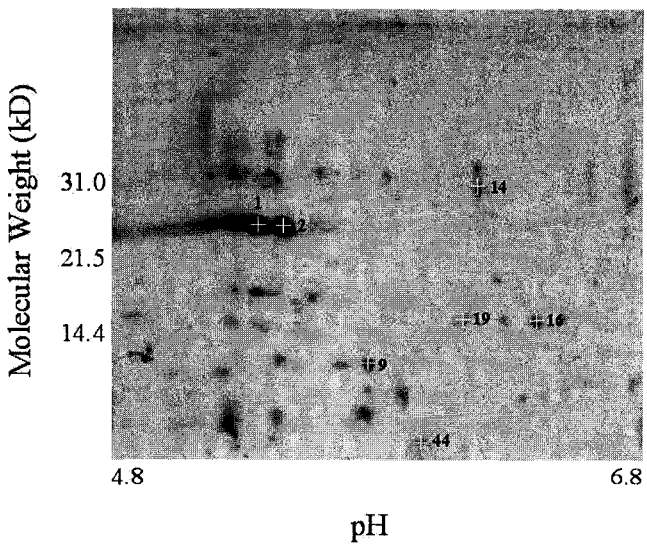


Figure 3.3: An example of a 2D map showing the locations of proteins identified in a gel

user can query the database through a Web interface using a client browser ("Database Search via Web Interface" in Figure 3.1). The Web interface provides the user several textual and graphical query methods. Secondly, our database can also be queried by another application via the Web services that are provided as part of the system ("Database Search via Web Services" in Figure 3.1). These two query methods greatly enhance the interoperability our system.

In addition to database queries, we also augment our database by maintaining a repository of tools for experimenting with various data analysis and data mining tasks, such as characteristic rule mining, sequential pattern analysis, association rules, etc. These tools provide users with the power of intelligently retrieving and analyzing data across a large array of heterogeneous data sets. For example, one tool that we are currently developing is the automatic identification of extracellular proteins from amino acid sequences. Adding a repository of tools to our database is of paramount importance for the long-term usefulness of our system. Similarly, the tools in the repository can be accessed by users through the Web interface or by other applications through the Web services.

Another novel aspect of our database is that it allows users to submit their new data and new tools. The data or tools submitted by users are maintained in a repository (see Figure 3.1). These data/tools are reviewed to make sure they do not contain inconsistencies or errors. If the data/tools pass the reviewing process, they are integrated into our database. In this way, our database acts as an "information hub" for biologists all over the world who are working on extracytosolic plant proteins and expedite information exchange and sharing among them.

3.3 Database Construction

Many existing 2D-PAGE databases use the *Make2ddb* package [29] to build a 2D-PAGE database on one's own Web server. The main focus of *Make2ddb* is on ease of use. However, we chose not to use it in our case for the following

reasons:

- *Make2ddb* uses text files rather than database management systems to manage the data, which might cause performance problems when the volume of data is large.

- The queries generated by *Make2ddb* are fixed, i.e., it only allows searching by description (DE or ID line), by accession number (AC line), by clicking on a spot, and by author (RA line). However, in our database, we want to allow users to submit more sophisticated queries.

- *Make2ddb* does not generate an API (Application Programming Interface) for the 2D-PAGE database it creates. That means other applications cannot communicate with the database easily. In our database, we provide an API using SOAP (Simple Object Access Protocol) technology[2] to allow inter-operations between applications.

We choose to use MySQL[3] as our database management system since it is a free and powerful relational database management system. The query processing is implemented in PHP[4], a scripting language especially suited for Web application development. The PHP and MySQL combination is cross-platform compatible and is commonly used for creating data-driven Web sites. The schema of the database is shown in Figure 3.4. Currently, the "spot" is implemented in text files for some technical issues.

The user can query the database in a variety of ways. The "quick text search" (Figure 3.5) is currently set up to retrieve entries that contain the specified keyword(s) in the "DE" line. The user can also query the database by clicking on a spot in a gel image (similar to other 2D-PAGE databases). In addition, if a user chooses "advanced text search", the attributes (text lines) to be searched can be explicitly specified.

[2]http://www.w3.org/TR/SOAP
[3]http://www.mysql.com/
[4]http://www.php.net/

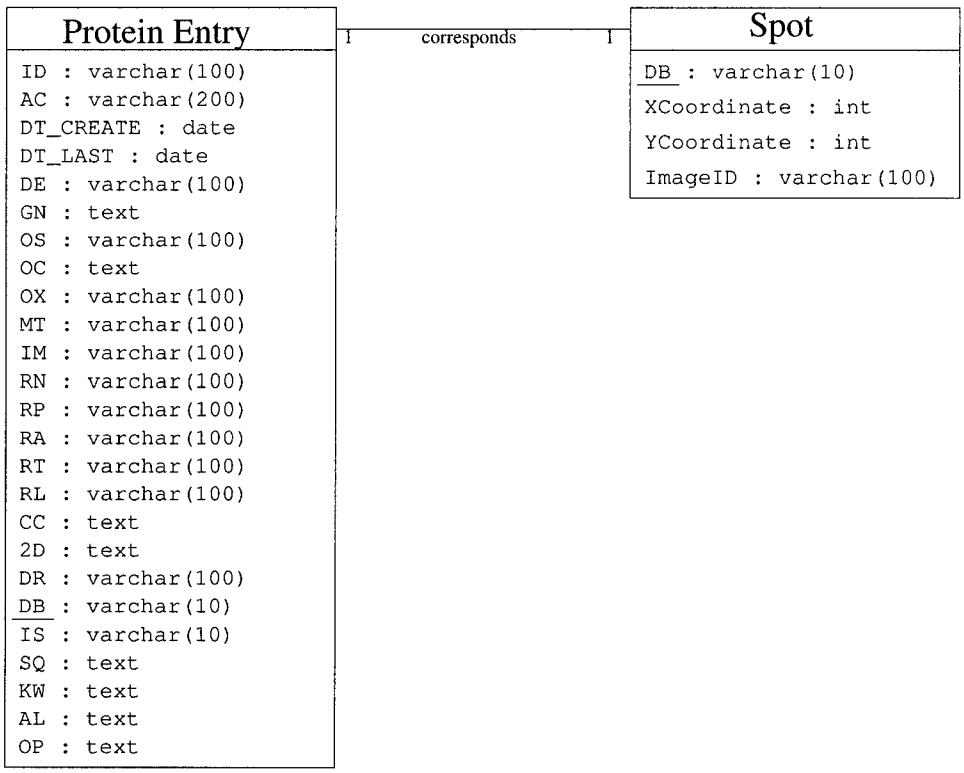


Figure 3.4: EPPdb database schema

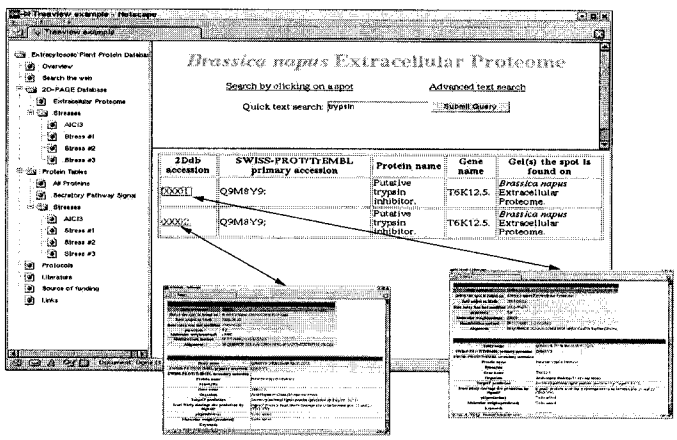


Figure 3.5: Quick text search

3.4 Web Services

Most other protein databases focus on building applications that are made globally available through a Web server, defining their user interfaces with HTML, and can be accessed using client browsers. The applications do not take advantage of the Internet to make the services available to a variety of clients. An example is SWISS-PROT. If other protein databases need to link to SWISS-PROT, they must generate pointers to the information related to SWISS-PROT entries. However, SWISS-PROT entries are dynamically generated HTML pages and their layouts could be inconsistent. If proper formats are not used, there is no way for other databases to directly exploit the information provided by SWISS-PROT entries in HTML format.

The idea of Web Services is instead of supplying information via dynamically generated user interfaces (HTML) that are fairly fixed and can only be consumed by client browsers, the server makes available a series of function calls over HTTP and uses XML as a message transfer format to be consumed by the clients. This gives much more freedom for the clients to use the services in whatever way they wish, since XML is a standard format for structured documents and data that is platform and language independent. In addition, if functions are called using standard HTTP-based protocols over the Internet, then the client that calls them can be located anywhere on the Internet. There are no restrictions on what platform it might be running on or in which language it should be written.

3.4.1 Introduction to Web Services

Web Services, in the most simplistic fashion, provide a mechanism of communication between two remote systems, connected through the network of the Web Services. For example, in case of different 2-DE database applications maintained by different groups in the world, people do not want to invest large amount of money developing software to bring these different applications together. By extending the applications as Web Services, the information systems of different applications can be linked. These applications can be ac-

cessed by using simple SOAP (see Section 3.4.3) messages over the normal HTTP Web protocol.

Web Service is not the first solution to such a problem. RMI, COM, CORBA, EDI, and ebXML also address the same problem. However, Web Service is different from the others in that this technology is based on the already existing and well-known HTTP protocol, and uses XML (see Section 3.4.2) as the base language. This makes it a very developer-friendly service system. However, most of the above-mentioned technologies such as RMI, COM, CORBA involve a whole learning curve. New technologies and languages have to be learned to implement these services. Also Web Services are based on a set of standardized rules and specifications, making the technology very portable. This is not the case with the technologies mentioned earlier [47].

The architecture for Web Services has the following components[5]:

- a standard way for communication.
- a uniform data representation and exchange mechanism. XML (Extensible Markup Language) is used extensively for this purpose.
- a standard meta language to describe the services offered, specifically a language called WSDL (Web Service Definition Language) has been developed for describing Web Services.
- a mechanism to register and locate Web Services-based applications.

Figure 3.6 shows the architecture of Web Services.

3.4.2 Extensible Markup Language (XML)

Web Service architecture involves many technologies. One critical technology is XML (eXtensible Markup Language). XML is a universally agreed markup meta-language primarily used for information exchange. A good example of a markup language is the HyperText Markup Language (HTML). The beauty

[5] www.w3.org/TR/ws-arch

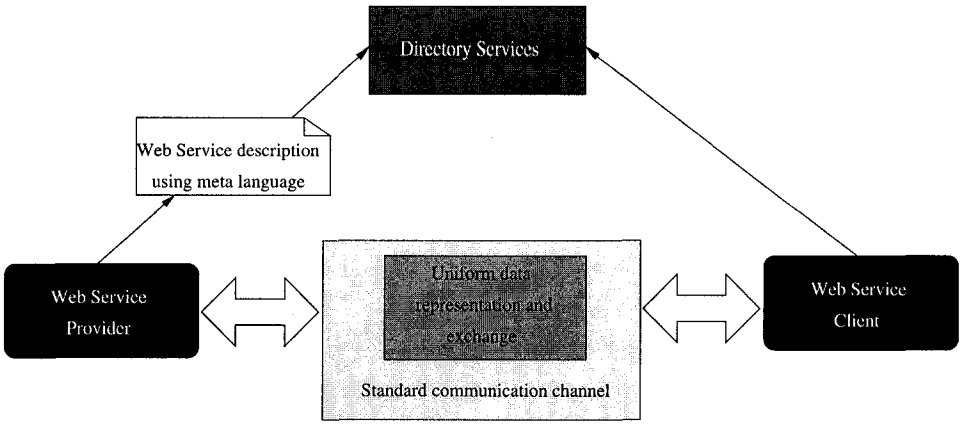


Figure 3.6: Architecture of Web Services

of XML lies in the fact that it is extensible. XML is a set of predefined rules (syntactical framework) that you need to follow when structuring your data. XML provides a standard and common data structure for sharing data between systems.

The following is an example of an XML document, representing the personal information and shift data for an employee in an organization.

```
<employee>
<shift id=''counter'' time=''8-12''>
  <phone id=''1''>
    <number>3444333</number>
  </phone>
</shift>
...
<home-address>
  <street>3434 Norwalk street</street>
  <city>New York</city>
  <state>NY</state>
</home-address>
</employee>
```

Also an example of XML document used in EPPdb is shown as follows:

```
...
```

```
<id xsi:type=''xsd:string''>
  081352; PRELIMINARY; 2DG.
</id>
<ac xsi:type=''xsd:string''>
  081352;
</ac>
<dt xsi:type=''xsd:string''>
  2003-05-30
</dt>
...
```

There are some XML languages developed or being developed for biological data, e.g., GAME[6] (Genome Annotation Markup Language), BIOML[7] (BIOpolymer Markup Language), BSML[8] (Bioinformatic Sequence Markup Language), AGAVE[9] (Architecture for Genomic Annotation, Visualization and Exchange), DAS[10] (Distributed Sequence Annotation System), ProML[11] (Protein Markup Language), PROXIML[12] (PROtein eXtensible Markup Language), etc. Most of these XML standards are developed individually by some organizations. Also they are usually designed for some particular biological data management tasks. Some of them are even still in the process of development and are not publicly accessible. To the best of our knowledge, there are currently no widely agreed upon XML standards available.

3.4.3 Simple Object Access Protocol (SOAP)

Another technology important to Web Services is SOAP (Simple Object Access Protocol). SOAP is a universally agreed on standard protocol for invoking the functions available in Web Services. With SOAP as an XML messaging

[6] www.bioxml.org/Projects/game/index.html
[7] www.bioml.com/BIOML/index.html
[8] www.labbook.com/products/xmlbsml.asp
[9] www.agavexml.org
[10] biodas.org
[11] cartan.gmd.de/promlweb
[12] www.cse.ucsc.edu/douglas/proximl

specification, Web Services enable developers to target a range of clients and build the services from local and remote resources.

A SOAP transaction begins with an application making a call to a remote procedure. The SOAP client script then encodes the procedure as an XML payload and sends it over the transport protocol to a server script. The server parses the request and passes it to a local method, which returns a response. The response is encoded as XML by the server and returned to the client, who parses the response and passes the result to the original function. So as long as other applications support XML and SOAP, they can communicate with Web Services and have the answer right away, instead of having to parse HTML documents [47].

3.4.4 Web Services Implementation

The search functions involved in our database are encapsulated into Web services. All Web services are implemented and published using NuSOAP[13], a freely available toolkit which provides a simple API for building Web services using Simple Object Access Protocol (SOAP) technology[14]. Other applications can remotely call these functions via XML. The implementation of Web services provides an appropriate solution for transparently integrating applications from heterogeneous sources. It is very easy for other applications to exchange information with EPPdb using Web Services. Appendix B shows an example of PHP code that is used to query EPPdb using Web Services.

3.5 Data Analysis Tools

One unique feature of our database is that it not only provides standard database query capabilities, but also provides domain-specific tools for experimenting with various data analysis and data mining methods. As described in Section 3.2, EPPdb allows users to submit data analysis and data mining tools. For a tool to be submitted to EPPdb, it must be able to run on the EPPdb

[13]http://dietrich.ganx4.com/nusoap/index.php
[14]http://www.w3.org/TR/soap

server and be able to send and receive XML messages. After the tool passes the reviewing process, the Web Services that it provides and relevant documentations on the formats of the XML messages are made publicly available. Other applications can communicate with this tool via XML.

Here we give an example of a data mining tool we have developed for our database, which is to predict extracellular proteins from amino acid sequences.

In order to function properly, proteins need to be localized at proper locations within the cell or transported to the extracellular environment. The set of locations depend on the type of a cell. For biologists, a very important question is to determine whether a given protein is an extracellular protein or non-extracellular protein (i.e., intracellular protein). This problem is a special case of protein subcellular localization prediction [12]. Most existing localization prediction methods use supervised learning algorithms to learn a classifier from a set of training data containing both intra- and extra-cellular protein sequences. When a new protein sequence comes in, the learned classifier is used to predict the correct label for the sequence.

The problem of extracellular protein prediction poses several challenges for most existing localization prediction algorithms: (i) the effectiveness of most existing algorithms is measured by overall accuracy. However, since extracellular proteins are extremely rare compared with intracellular proteins (less than 1%), predicting every protein as intracellular protein can achieve very high accuracy level of 99%; (ii) biologists are usually interested in those tools that provide some explanations of the prediction, i.e., they want a classifier to let them know why a protein is or is not predicted as extracellular. Many existing algorithms (e.g., Artificial Neural Networks, Support Vector Machines) are "black box" techniques, in the sense that they predict the class label for a given protein sequence without providing any easily interpretable justification. Even if the prediction is correct, biologists may be hesitant in using such non-transparent tools. If the prediction is incorrect, biologists are given no hints to identify the locations in a classifier's reasoning process that might cause the misclassification.

The rest of this thesis elaborates on the algorithms that we have developed

to solve this problem. We use data mining techniques to develop extracellular protein predictors based on the subsequences that appears frequently in extracellular proteins.

3.6 Summary

In this chapter, we have described in detail the design and implementation of various functionalities of EPPdb. Like most other 2-DE database, EPPdb provides textual and graphical query capabilities that allow biologists to populate and query the database. An novel aspect of EPPdb is that it provides an API allowing other applications to access this database as a Web service. In addition, we augment EPPdb with a repository of tools to be used in data analysis and data mining tasks. The following chapters will describe in detail one particular tool we have developed, which is to predict extracellular proteins from amino acid sequences.

Chapter 4

Work Related to Extracellular Protein Prediction

This chapter describes the problem of extracellular protein prediction and reviews existing approaches. This chapter is organized as follows. Section 4.1 provides some biological background and introduces the problem of protein subcellular localization prediction, a more general area in which our problem of extracellular protein prediction falls. Sections 4.2.2, 4.2.1 and 4.2.3 review some of the existing methods. Section 4.2.5 points to some limitations of existing methods for our particular problem. Section 4.4 summarizes this chapter.

4.1 Introduction to Biological Background

This section provides the biological background of our problem. More detailed information about the biological background can be found in [33, 40].

Proteins are the molecules that accomplish most of the functions of the living cell. All proteins are composed of linear sequences of smaller molecules called amino acids. Such a sequence is called the primary structure of a protein. There are twenty naturally occurring amino acids (Table 4.1). Long proteins may contain a chain of as many as 4500 amino acids. Finding the proteins that make up a creature and understanding their functions is the foundation of explanation in molecular biology [33].

With the introduction of large-scale sequencing, biologists have accumu-

	name	Three-letter code	One-letter code
1	Alanine	Ala	A
2	Cysteine	Cys	C
3	Aspartic Acid	Asp	D
4	Glutamic Acid	Glu	E
5	Phenylalanine	Phe	F
6	Glycine	Gly	G
7	Histidine	His	H
8	Isoleucine	Ile	I
9	Lysine	Lys	K
10	Leucine	Leu	L
11	Methionine	Met	M
12	Asparagine	Asn	N
13	Proline	Pro	P
14	Glutamine	Gln	Q
15	Arginine	Arg	R
16	Serine	Ser	S
17	Theronine	Thr	T
18	Valine	Val	V
19	Tryptophan	Try	W
20	Tyrosine	Tyr	Y

Table 4.1: The twenty naturally occurring amino acids

lated an immense number of raw biological sequences that are publicly available. In order to better understand the functions and structures of these protein sequences, a vitally important problem facing the biology community is to classify these sequences into different families based on the properties of the sequences, such as functions, structures, etc.

Protein subcellular localization is a key functional characteristic of proteins. In order to execute a common physiological function, proteins must be localized in the same cellular compartment. Proteins may be localized at various locations within the cell or be transported to the extracellular space. The process through which proteins are routed to their proper subcellular localizations is called subcellular protein sorting. Protein sorting is the simplest in gram positive prokaryotes (a kind of single cell organism), where proteins are only directed to the cytoplasm, the plasma membrane, the cell wall, or secreted to the extracellular space. Gram negative protein localization sites include the cytoplasm, the inner membrane, the periplasm, the outer membrane, and the extracellular space. Subcellular localizations in eukaryotic proteins are much more complex due to the presence of membrane-bound organelles. The major location sites for eukaryotic proteins include the plasma membrane, the nucleus, the mitochondria, the peroxisome, the endoplasmic reticulum, the Golgi apparatus, the lysosome, the endosome, and others (such as chloroplasts, vacuoles, and the cell wall in plant cells). Table 4.2 shows an ontology across five different taxonomic categories based on the PSORT ontologies [50].

The subcellular localization of a protein plays an important role with regard to its function. Knowledge of subcellular localization can provide valuable information concerning its possible functions. It can also help in analyzing and annotating sequences of hypothetical or known gene products. In addition, it can influence the design of experimental strategies for functional characterization [19, 48].

Since the number of sequences entering into data banks has been rapidly increasing, it is time consuming and costly to approach this problem of predicting the subcellular localization of a protein entirely by performing various biological experimental tests. In view of this, it is highly desirable to develop

Category	Subcellular Localizations
Animal	nuc, end, gol, mit, pex, lys, cyt, mem, ext
Plant	nuc, end, gol, mit, pex, chl, vac, cyt, mem, ext
Fungi	nuc, end, gol, mit, pex, vac, cyt, mem, ext
Gram-positive bacteria	cyt, wal, mem, ext
Gram-negative bacteria	cyt, inn, per, wal, out, ext

Table 4.2: The ontologies across five different taxonomic categories. Abbreviations for localizations: nuc (nuclear), end (endoplasmic reticulum), gol (golgi), mit (mitochondria), pex (peroxisomal), lys (lysosomal), cyt (cyoplasmic), mem (membrane), ext (extracellular), chl (chloroplast), vac (vacuole), inn (inner membrane), per (periplasmic), wal (cell wall), out (outer membrane).

some algorithms to rapidly predict the subcellular localizations of proteins.

In our project, we are particularly interested in identifying those proteins that are secreted to the extracellular environment (called *extracellular proteins*), versus proteins localized at various locations within the cell (called *intracellular proteins*). As described in Section 3.1, EPPdb is built for extracytosolic plant proteome. A tool that can automatically identify whether a protein is intracellular or extracellular is of great interest for biologists in the extracytosolic plant community.

The following sections (4.2.1, 4.2.2, 4.2.4 and 4.2.3) review the current approaches of protein subcellular localization prediction. Section 4.2.5 points out some limitations of current methods and explain why new algorithms are needed.

4.2 Work Related to Protein Subcellular Localization Prediction

4.2.1 Prediction Based on N-terminal Sorting Signals

Subcellular protein sorting (see Section 4.1) is a fundamental aspect of cellular life. In many cases, sorting depends on "signals" that can be identified by looking at the primary structure of a proteins [53]. Previous work [61, 66] has shown that when the final destination is the mitochondrion, the chloroplast, or the secretory pathway, sorting usually relies on the presence of an

N-terminal targeting sequences that can be recognized by a translocation machinery. Work has been done on identifying individual sorting signals, i.e., secretory signal peptides (SPs), mitochondrial targeting peptides (mTPs) and chloroplast transit peptides (cTPs) [53].

Signal peptides (SPs) are N-terminal peptides that control the entry to the general secretory pathway, which is a mechanism for protein secretion found in both eukaryotic and prokaryotic cells [56]. SPs generally consist of three regions (Figure 4.1): a positively charged n-region, a hydrophobic h-region, and a polar c-region leading up to the signal peptide cleavage site. Signal peptides are often cleaved off the mature proteins upon arrival at the subcellular destination. The most well-conserved motif[1] of SPs is the presence of a small and neutral amino acid at positions -3 and -1 relative to the cleavage site [76, 77]. Neilsen *et al.* [54, 55] developed a method (SignalP) for the identification of signal peptides and their cleavage sites based on neural networks trained on separate sets of prokaryotic and eukaryotic sequences. Their method does not provide any explanation for the prediction results. Nielsen and Krogh [56] developed an hidden Markov model of signal peptides (SignalP-HMM), which contains submodels for the N-terminal part, the hydrophobic region, and the region around the cleavage site. For known signal peptides, the model can be used to assign objective boundaries between these three regions. Other methods used for predicting subcellular localizations based of signal peptides include discriminant function analysis [13, 12], weight matrices [17], and others.

In mitochondrial targeting peptides (mTPs), Arg (R), Ala (A) and Ser (S) are over -represented, while negatively charged amino acid residues (Asp (D) and Glu (E)) are rare. Only weak consensus sequences have been found, the most prominent of which is a conserved Arg(R) in position -2 and -3 relative to the mitochondrial processing peptidase (MPP) cleavage site [21]. Furthermore, mTPS are believed to form an amphiphilic α-helix [27, 62, 79].

Chloroplast transit peptides (cTPs) are the N-terminal presequences found in most nuclearly encoded chloroplast proteins that direct them to the chloro-

[1] a short conserved region in a protein sequence

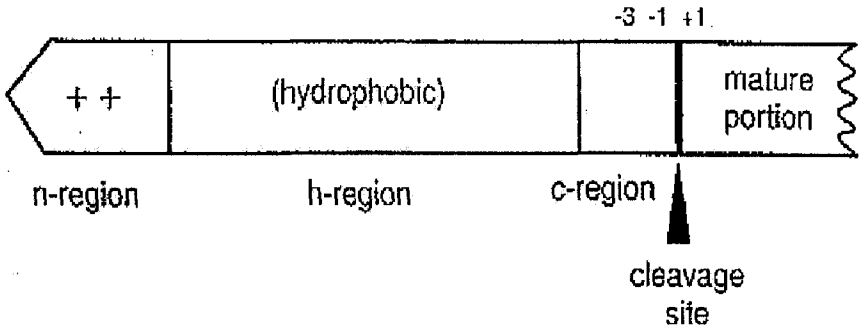


Figure 4.1: Structure of signal peptide [51]

plast stroma [70]. The secondary structure of chloroplast transit peptides is not well characterized. Still, cTPs have a few distinguishing features, such as low content of acidic residues and rich in hydroxylated residues [78]. Emanuelsson *et al.* [20] developed a neural network based method (ChloroP) for identifying chloroplast transit peptides and their cleavage sites. ChloroP achieved an accuracy of 88% in classifying sequences as transit peptides or nontransit peptides. Cleavage sites are predicted using a scoring matrix derived by an automatic motif-finding algorithm. About 60% of the known cleavage sites were predicted to within +/-2 residues from the cleavage sites given in SWISS-PROT.

Emanuelsson *et al.* [21] also proposed an integrated prediction system (TargetP) for subcellular localization using neural networks. TargetP can be used to differentiate four subcellular localizations (extracellular, mitochondrial, chloroplast and "other") based on the individual sorting signal predictions (signal peptides for extracellular, mitochondrial transit peptides for mitochondria, chloroplast transit peptides for chloroplast, and no sorting signal for "other").

The advantage of predicting subcellular localizations based on N-terminal sorting signals is that it can recognize cleavage sites in the sorting signals. However, these methods would be inaccurate when the signals are missing or only partially included [32]. In addition, the known signals are not general enough to cover the resident proteins in each organ [51].

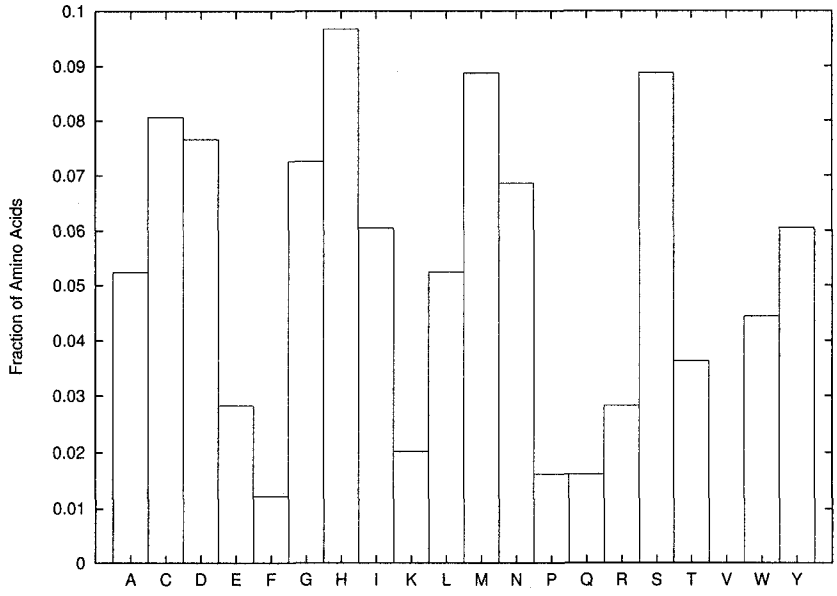


Figure 4.2: An example of amino acid composition represented by a histogram

4.2.2 Prediction Based on Amino Acid Composition

The amino acid composition of a protein sequence refers to the relative frequencies of twenty different amino acids in this sequence. Figure 4.2 shows an example of the amino acid composition of a plant protein sequence represented by a histogram. Nakashima and Nishikawa [52] showed that intracellular and extracellular proteins differ in their amino acid composition. Andrade *et al.* [3] indicated that protein subcellular localizations correlate better with the surface composition. Different methods, including statistical methods, neural network, and support vector machines, have been proposed for protein subcellular localization based on amino acid composition information.

Nakashima and Nishikawa [52] proposed a statistical analysis-based algorithm to discriminate between intracellular and extracellular proteins by amino acid composition and residue-pair frequencies, which correctly classify 88% of intracellular and 84% of extracellular proteins. Cedano *et al.* [11] proposed a statistical methods using the Mahalanobis distance to extend the discriminative class from two to five, i.e., extracellular, integral membrane, anchored membrane, intracellular and nuclear. They proposed an algorithm called ProtLock [11] to improve the prediction quality of protein subcellular locations,

75% of 200 proteins from SWISS-PROT are correctly classified. Chou and Elrod [12] proposed a covariant discriminant algorithm to further improve the accuracy to 79.9%. They have shown that their results are better than other methods based on statistical analysis.

NNPSL [60] is a amino acid composition based prediction system constructed using neural networks. It computes the fraction of each of the twenty amino acids (see Figure 4.3), then uses these fractional numbers for the input units of neural networks. Output units are used for predictions that distinguish between possible locations. It deals with prokaryotic and eukaryotic sequences separately. For prokaryotic sequences, three possible locations (cytoplasmic, periplasmic and extracellular) can be predicted. For eukaryotic sequences, four locations (nuclear, cytoplasmic, mitochondrial, extracellular) can be predicted. NNPSL achieves an overall accuracy of 66% on eukaryotes (excluding plants) and 81% on prokaryotes.

Hua and Sun [32] applied support vector machine (SVM) to build a prediction system called SubLoc. The SVM approach is a state-of-the-art method for the binary classification problem. In order to handle multi-class problems, they built separate k SVM classifiers for each of the k possible classes, where the ith SVM was trained with all of the samples in the ith class with positive labels and all other samples with negative labels. Finally an unknown sample is classified into the class that corresponds to the SVM with the highest prediction confidence. SubLoc achieves an accuracy of 91.4% on prokaryotes and 79.4% on eukaryotes.

4.2.3 Prediction Based on Lexical Analysis

The SWISS-PROT and many other protein databases contain textual annotations on the subcellular localizations for many proteins. Another approach is to infer the subcellular localization of an unknown proteins by lexical analysis of "keywords", which are the words extracted from certain attribute lines of the annotated protein entries. This is similar to the problem of "Text Categorization". Text Categorization is the problem of assigning predefined categories to unseen documents. A lot of methods have been developed for text categoriza-

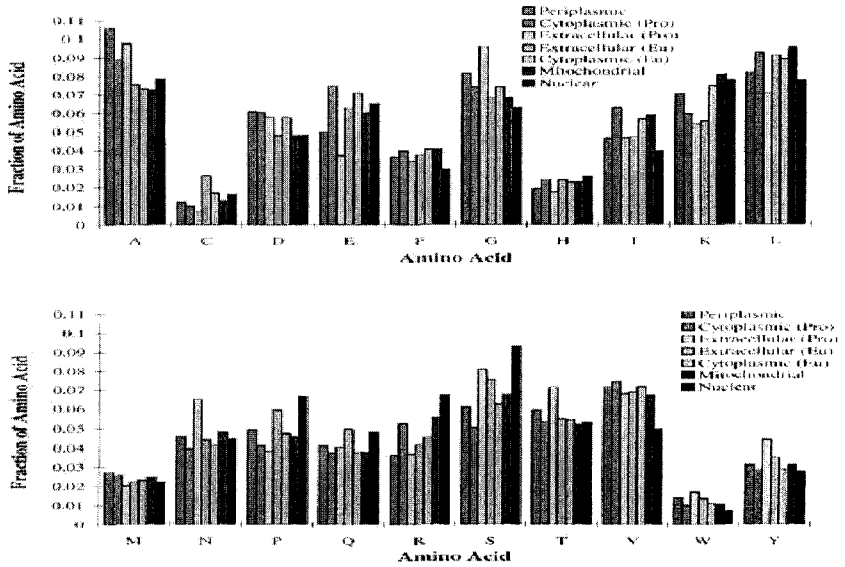


Figure 4.3: The overall differences in amino acid composition between all groups [43]

tion, such as nearest neighbor classifiers [84], multivariate regression models [83, 68], probabilistic Bayesian models [41], symbolic rule learning [6] etc.

LOCkey [49] is an example of protein subcellular localization prediction based on lexical analysis. The algorithm of LOCkey is shown in Figure 4.4.

- LOCkey firstly compiled a data set with proteins of experimentally known localization. For each protein sequence, a homology search tool BLAST [2] was used to find the homologous protein sequences. Protein homology refers to the similarity attributed to descent from a common ancestor. If two proteins are homologous, they are likely to share common characterizations.

- It merged keywords from SWISS-PROT for the proteins in this set and their homologous proteins from the "keyindex" file.

- A data set of binary vectors (called "trusted vector set") was built for these keywords that represented the presence of a certain keyword by 1 and the absence by 0. In order to reduce the dimensionality of feature space, only keywords with "above random" (measured by "information content") classifying ability based on an entropy and normalized entropy

cut-off are retained.

- To infer the localization of a protein U of unknown localization, LOCkey first retrieved all the keywords for U from SWISS-PROT that matched the "trusted vector set" of informative keywords. Thus a vector $V(U)$ that had the same dimension as the vectors in the "trusted vector set" was retrieved. Next, all possible alternatives (called "sub-vectors") to $V(U)$ are generated, for which one or many 1's were flipped to 0's. For example, for a protein with 3 keywords, it generated $2^3 - 1 = 7$ sub-vectors $V'(U)$: 111, 110, 101, 011, 100, 010 and 001.

- Finally the best matching vector is found based on entropy criteria methods for inferring the subcellular localization of the query sequence.

PA-sub [43] is another protein subcellular localization prediction system based on lexical analysis. The algorithm of PA-sub is shown in Figure 4.5 Different from LOCkey which used simple entropy criteria method to infer the subcellular localization of the query sequence. The authors of [43] compared several sophisticated machine learning algorithms, including k nearest neighbor, naive Bayes, tree-augmented naive Bayes, artificial neural network and support vector machine. In order to remove the "trivial" keywords, PA-Sub used a wrapper model to select those informative keyword based on information content.

4.2.4 Prediction Based on Integrative Approaches

Some methods for predicting protein subcellular localization take an integrative approach by combining several different methods. PSORT [50] is probably the most robust tool for predicting subcellular localization. It is an expert system developed to distinguish between seventeen different subcellular localizations in eukaryotes. The system integrates numerous prediction programs and statistical methods to make a prediction.

Recently, an updated version of PSORT, called PSORT-B [25], was presented for the prediction of protein subcellular localization for Gram-negative

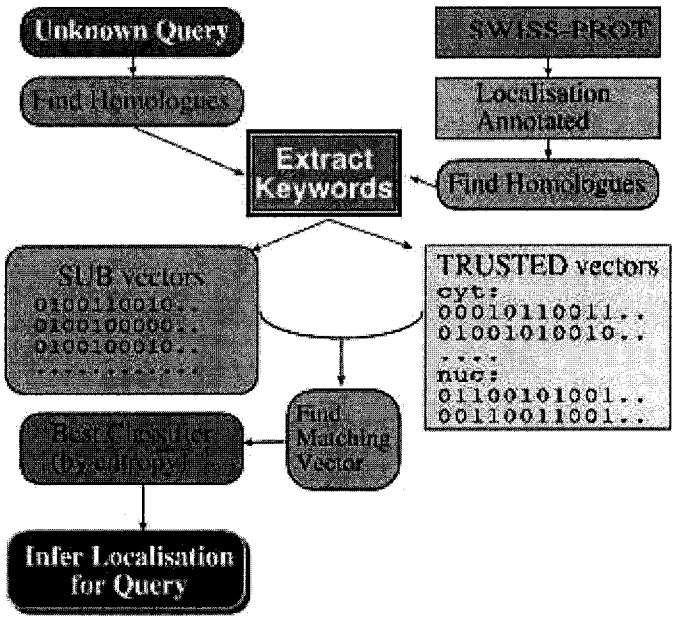


Figure 4.4: LOCkey algorithm [49]

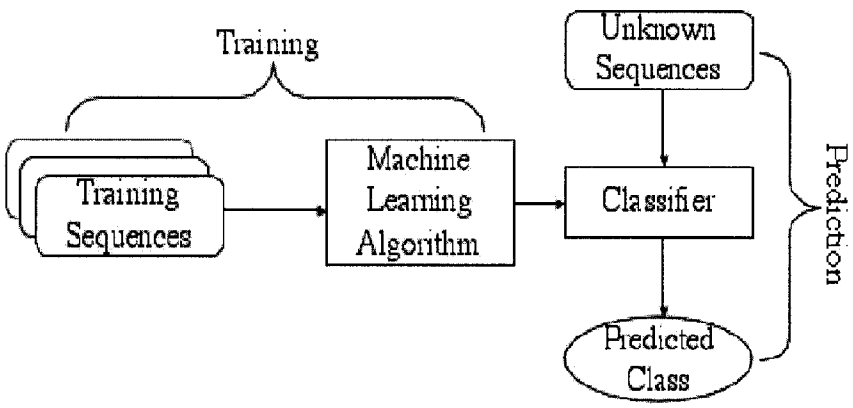


Figure 4.5: PA-Sub builds a classifier using machine learning algorithms [43]

bacteria. Given a Gram-negative bacteria protein sequence with unknown subcellular localization, PSORT-B combines several different methods, including homology analysis, identification of sorting signals and other motifs, and machine learning algorithms into an expert system for the prediction of five subcellular localizations. A query protein undergoes each of the analyses separately and the results are combined together.

4.2.5 Challenges and Limitations of Existing Methods

In our project, we are particularly interested in those proteins that are secreted from the cell (i.e., extracellular proteins). We study the problem of identifying extracellular proteins from sequence information alone. As mentioned in Section 3.1, EPPdb is an on-line database devoted to the extracytosolic plant proteome. The identification of extracellular proteins are of particular interest to potential users of the database. However, most of the existing approaches are not suitable for this particular application.

Firstly, most of the existing systems use overall accuracy to measure how good the prediction is. However, in our particular case, overall accuracy is not an appropriate measurement. The reason is due to the fact that most of the proteins are intracellular proteins. Extracellular are extremely rare among all the proteins (e.g., less than 5%). In this case, high overal accuracy (higher than 95% can be easily achieved by predicting every protein as intracellular. But this kind of classifier does not provide any valuable information for the user.

Secondly, since most of the existing prediction tools aim to predict all the subcellular locations, they usually use a training dataset that contains proteins with all the possible class labels. However, in our application, the dataset that we will eventually get does not necessarily distinguish different locations among intracellular proteins.

Thirdly, biologists are very interested in those subsequences that discriminate extracellular proteins from intracellular proteins. Approaches based on amino acid composition or lexical analysis obviously do not provide such insights for biologists to make further analysis. Approaches based on N-terminal

signals can provide some insights, since they can identify the signal peptides. However, since those approaches are based on artificial neural networks, people have no clues on how these signal peptides are identified.

In this dissertation, we apply data mining techniques to build classifiers based on frequent subsequences to address this problem.

4.3 Work Related to Mining Sequential Data

The discovery of patterns in sequential data has been an important problem in the general area of data mining and knowledge discovery. Since protein primary sequences are made up of amino acid sequences, our work is related to data mining in sequential data. This section reviews some existing problems and different solutions in this field.

4.3.1 Mining Sequential Patterns

The problem of sequential pattern mining was introduced in [1]: *Given a set of sequences, which each sequence consists of a list of elements and each element consists of a set of items, and given a user-specified min-support threshold, sequential pattern mining is to find all of the frequent subsequences, i.e., the subsequences whose occurrence frequency in the set of sequences is no less than the min-support.* In [86, 87], a unique *transaction time* is associated with each element in a sequence. The *length* of a sequence is define as the number of instances of items in the sequence. A sequence with length l is called $l-sequence$. A sequence $\alpha =< a_1 a_2 ... a_n >$ (where $a_1, a_2, ..., a_n$ are elements of α) is called a *subsequence* of another sequence $\beta =< b_1, b_2, ..., b_m >$ (where $b_1, b_2, ..., b_m$ are elements of β), i.e., $\alpha \sqsubseteq \beta$, if there exist integers $1 \leq j_1 < j_2 < ... < j_n \leq m$ such that $a_1 \subseteq b_{j1}, a_2 \subseteq b_{j2}, ..., a_n \subseteq b_{jn}$. And β is called a $super - sequence$ of α.

For example, Table 4.3 shows a *sequence database* with four sequences. The set of *items* in the database is $\{A, B, C, D, E, F, G, H\}$. The first sequence (with Sequence_id 1) has four elements: (CD), (ABC), (ABF) and (ACDF). It is also a 12-sequence since it contains 12 instances of items. Item A happens 3

Sequence_id	Transaction-Time	Items
1	10	C D
1	15	A B C
1	20	A B F
1	25	A C D F
2	15	A B F
2	20	E
3	10	A B F
4	10	D G H
4	20	B F
4	25	A G H

Table 4.3: A sequence database [87]

times in this sequence. However the sequence <(CD) (ABC) (ABF) (ACDF)> is considered to contribute only one to the *support* of $< A >$. Sequence <(C) (AB) (AF)> is a *subsequence* of <(CD) (ABC) (ABF) (ACDF)>. If *min-support=2*, sequence $s =< (D)(BF)(A) >$ is a *sequential pattern* since both sequences 1 and 4 contain sequence s.

Many studies have been conducted on finding sequential patterns from a database of sequences. In general, these algorithms can be classified into three categories: (1) Apriori-based, horizontal formatting method, such as GSP [71]; (2) Apriori-based, vertical formatting method, such as SPADE [87], (3) projection-based pattern growth methods, such as PrefixSpan [57]. Below we briefly describe these three representative algorithms.

GSP Algorithm

GSP is based on the *apriori* heuristic, which says *a sequence can not be frequent if any of its subsequence is not frequent.* The algorithm is shown in Figure 4.6. It makes multiple scans over the database. In the first scan, all frequent 1-sequences are found (line 2) . From frequent 1-sequences, a set of candidate 2-sequences are generated (line 5). Another database scan is used to calculate the support of each candidate 2-sequences, those candidates with support greater than the minimum support threshold form the set of frequent 2-sequences (line 8). Similarly, frequent 3-sequences are obtained based on frequent 2-sequences. This process is repeated until no more frequent sequences are found. Before we

Frequent 3-Sequences	Candidate 4-sequences	
	after join	after pruning
$< (1,2)(3) >$	$< (1,2)(3,4) >$	$< (1,2)(3,4) >$
$< (1,2)(4) >$	$< (1,2)(3)(5) >$	
$< (1)(3,4) >$		
$< (1,3)(5) >$		
$< (2)(3,4) >$		
$< (2)(3)(5) >$		

Table 4.4: A example of candidate generation in GSP [71]

describe these two steps, we first define the notion of *contiguous subsequence.* Given a sequence $s =< s_1 s_2 ... s_n >$, a subsequence c is called a *contiguous subsequence* of s if any of the following conditions hold: (1)c is obtained from s by dropping an item from either s_1 or s_n; (2)c is obtained from s by dropping an item from an element s_i, which contains more than one item; (3)c is obtained from s by executing the above two operations multiple times.

There are two main steps in GSP algorithm:

- Candidate Generation: Given the set of frequent $(k-1)$-sequences F_{k-1}, the candidates of k-sequences C_k are generated by joining F_{k-1} with itself. Then a pruning phase is applied to eliminate those candidates, of which at least one contiguous $(k-1)$-subsequence is not frequent. Table 4.4 shows an example of generating C_4 from F_3.

- Candidate Counting: For fast counting, candidate sequences are stored in a hash-tree data structure. To find all candidates contained in an input sequence s, conceptually all k-subsequences are generated. A search is made in the hash tree for each of such subsequence. If a candidate matches a subsequence in the hash tree, its count of that subsequence is incremented.

SPADE Algorithm

SPADE is a different algorithm for mining sequential pattern proposed by Zaki [86, 87]. In SPADE, the database of sequences is transformed into a vertical layout. In this vertical layout, an *id-list* is created for each item. The *id-list*

Algorithm GSP
Input: a sequence database D and minimum support threshold δ
Output: the set of all frequent subsequences
1.Function GSP(D,δ)
2. F_1={frequent 1-sequences};
3. F=F_1;
4. **for** (k=2; $F_{k-1} \neq \varnothing; k++$) **do**
5. C_k=set of candidate k-sequences;
6. **for** all sequences s in database D **do**
7. increment count of all $\alpha \in C_k$ contained in s;
8. $F_k = \{\alpha \in C_k | \alpha.sup \geq \delta\}$;
9. **return** $\bigcup_i F_i$;

Figure 4.6: GSP Algorithm

A	
CID	TID
1	15
1	20
1	25
2	15
3	10
4	25

B	
CID	TID
1	15
1	20
2	15
3	10
4	20

C	
CID	TID
1	10
1	25
4	10

D	
CID	TID
1	20
1	25
2	15
3	10
4	20

Figure 4.7: An example of vertical layout of sequence database

is a list of sequence id (*cid*) and time stamp identifier (*tid*). A *(cid, tid)* pair associated with an item records in which sequence this item occurs, and where it occurs in that sequence. Figure 4.7 shows the *id-lists* for the 1-sequence of the sequence database in Table 4.3.

Frequent 1-sequences can be found by a single scan of the *id-lists* of 1-sequences. After that, frequent k-sequences are generated be intersecting the id-lists of all distinct pairs of frequent $(k-1)$-sequences and checking the cardinality of the resulting *id-lists* for k-sequences against *min_sup*. Figure 4.8 shows the shows the process of id-list intersection.

Both breadth first and depth first search can be used to search the lattice shown in Figure 4.8 to find the frequent subsequences. Given a limited amount

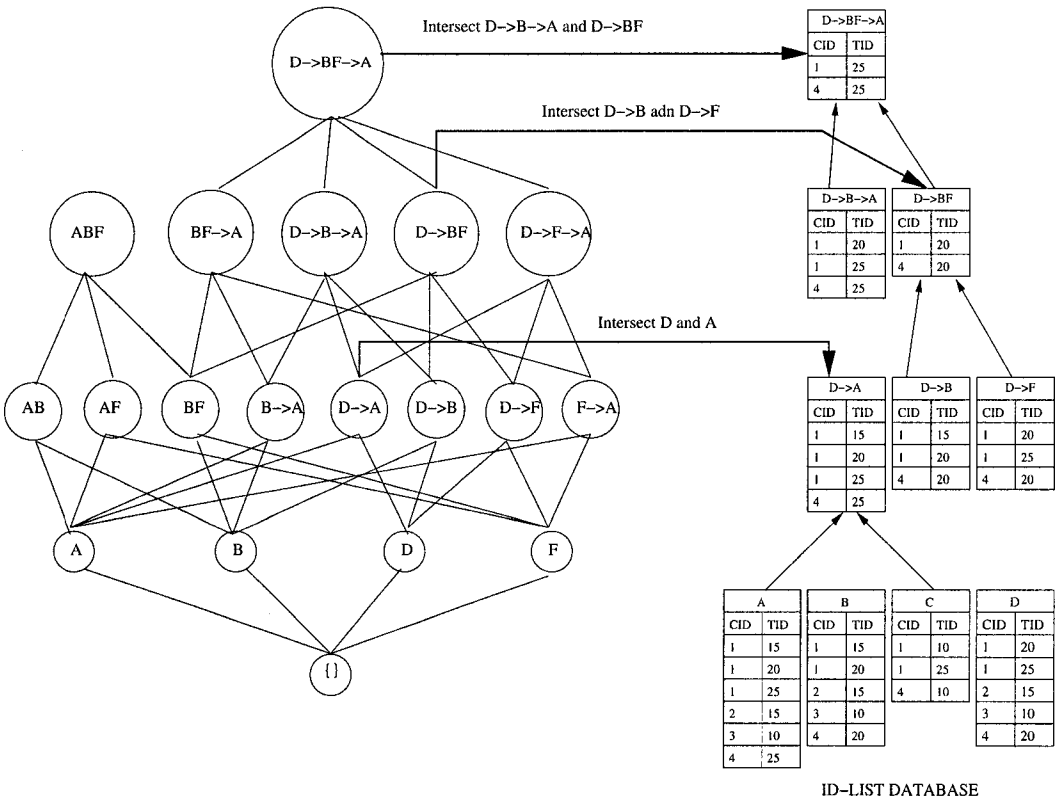


Figure 4.8: Computing Support via Id-list Intersections [86]

of main memory, the intermediate id-lists introduced in the lattice of Figure 4.8 usually cannot be fit in memory. The author of SPADE proposed to decompose the search space into equivalence classes such that each equivalence class can be processed independently, so at any time, only one equivalence class need to be stored in the main memory.

PrefixSpan Algorithm

PrefixSpan [57] is an algorithm that exploits prefix-projection in sequential pattern mining. Assuming items in any element are listed alphabetically, a sequence $\beta =< b_1 b_2 ... b_m >$ is called a *prefix* of another sequence $\alpha =< a_1 a_2 ... a_n >$ $(m \leq n)$ if and only if $(1) a_i = b_i$ for $(i \leq m-1)$; $(2) b_n \subseteq a_n$; (3) all items in $(a_n - b_n)$ are alphabetically after those in b_n. Given sequences α and one of its prefix β, a subsequence α' of α is called a *projection* of α w.r.t. β if and only if (1) α' has prefix β; (2) there exists no proper super-sequence α'' of α'.

Algorithm PrefixSpan
Input: a sequence database S and minimum support threshold δ
Output: the complete set of sequential patterns
Method: call PrefixSpan(<>, 0, S)
Subroutine PrefixSpan($\alpha, l, S|_\alpha$)
Parameters: α: a sequential pattern; l: the length of α; $S|_\alpha$: the α-projected database, if $\alpha \neq \varnothing$; otherwise, the sequence database S.
Method:

1. Scan $S|_\alpha$ once, find the set of frequent items b such that
   (a) b can be assembled to the last element of α to form a sequential pattern; or
   (b) $< b >$ can be appended to α to form a sequential pattern.
2. For each frequent item b, append it to α to form a sequential pattern α', and output α';
3. For each α', construct α'-projected database $S|_{\alpha'}$, and call PrefixSpan($\alpha', l+1, S|_{\alpha'}$).

Figure 4.9: PrefixSpan Algorithm

Similar to SPADE, frequent 1-sequences are found by a single scan of the database. Then the database is *projected* into several subsets, each has a distinct frequent 1-sequence as its prefix. Sequences containing $< a >$ are projected with respect to $< a >$ to form the $< a >$*-projected database.* All the frequent 2-sequences with prefix $< a >$ can be found by a single scan in $< a >$-projected database. In particular, given a prefix α of length k, all frequent $(k+1)$-sequences with prefix α can be found by a scan of α-projected database. This process ends when no more frequent sequences can be found. The algorithm of PrefixSpan is shown in Figure 4.9.

Two techniques are used to optimize the process. A *bi-level projection* scheme is proposed to reduce the number and the size of projected databases. A *pseudo-projection* is proposed to reduce the cost of projection when a projected database can be held in main memory.

4.3.2 Sequential Pattern Mining with Constraints

In many real applications, people are interested in finding sequential pattern with certain constraints. For example, when mining market-based sequential patterns, users often want to place a bound on the maximum distance between the occurrence of adjacent pattern elements in a data sequence [71].

One possible constraint in sequential pattern mining is in the form of regular expression (RE). A RE constraint $\mathcal{R}$ is specified as a RE over the alphabet of the sequence items using the established set of RE operators, such as disjunction (|) and Kleene closure (*) [26]. Results from complexity theory shows that given a RE $\mathcal{R}$, there exists a *deterministic finite automata (DFA)* $\mathcal{A}_{\mathcal{R}}$ such that $\mathcal{A}_{\mathcal{R}}$ accepts exactly the same language generated by $\mathcal{R}$. For example, Figure 4.10 shows an DFA that corresponds to the RE 1*(2 2 | 2 3 4 | 4 4). Garofalakis *et al.* [26] developed a method called SPIRIT for sequential pattern mining in the presence of user-specified regular expression (RE) constraints. They consider only sequences of simple items, i.e., every element in the sequence has only one item. Given an DFA (or an equivalent RE), a sequence α is valid if it can be accepted by the DFA that is given. For example, given the DFA in Figure 4.10, $< 1122 >$ and $< 234 >$ are examples of valid sequences. The problem addressed in [26] is: given a database of (simple) sequences $\mathcal{D}$, a user-specified minimum support threshold, and a user-specified RE constraint $\mathcal{R}$ (or an equivalent DFA), find all frequent and valid sequential patterns in $\mathcal{D}$.

A algorithmic framework called SPIRIT is proposed in [26] for solving this problem. The framework is similar to the apriori strategy used in GSP. The crucial difference is that they use relaxed constraints that have nice properties (e.g., anti-monotonicity) to filter out some candidates in the early stage.

Pei *et al.* [58] presented seven categories of constraints based on the semantics and the forms of the constraints.

Constraint 1. An *item constraint* specifies what are the particular individual or groups of items that should or should not be in the patterns.

Constraint 2. A *length constraint* specifies the requirement on the length of

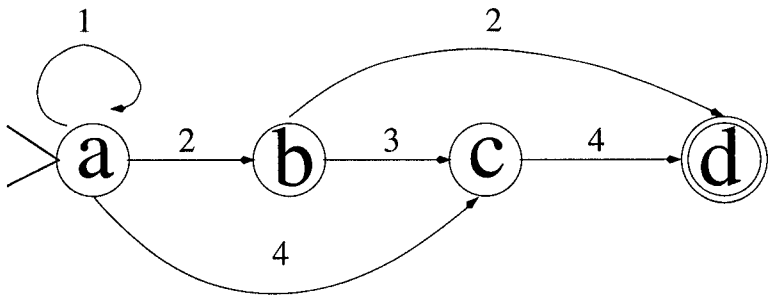


Figure 4.10: Computing Support via Id-list Intersections [26]

the patterns, where the length could be the number of items or the number of elements (refer to the definitions in Section 4.3.1).

Constraint 3. A *super-pattern constraint* is to find patterns that contain a particular set of patterns as sub-patterns.

Constraint 4. An *aggregate constraint* is the constraint on an aggregate of items in a pattern, where the aggregate function could be `sum`, `avg`, `max`, `min`, `standard deviation`, etc.

Constraint 5. A *regular expression constraint* is a constraint expressed as a regular expression over the set of items using the established set of regular expression operators, such as disjunction and Kleene closure.

Constraint 6. A *duration constraint* is defined only in the sequence database where each element in the sequence has a time-stamp associated with it. This constraint requires that the pattern appears frequently in the sequence database such that the time-stamp difference between the first and the last element in the pattern must be longer or shorter than a given period.

Constraint 7. A *gap constraint* is defined only in the sequence database where each element in the sequence has a time-stamp associated with it. This constraint requires that the pattern appears frequently in the sequence database such that the time-stamp different between every two adjacent transactions must be longer or shorter than a given gap.

Pei *et al.* summarize some commonly used constraints for sequential pattern mining in Table 4.5 based on the notion of *monotonicity, anti-monotonicity* and *succinctness.* A constraint C_M is **monotonic** if a sequence α satisfying C_M implies that any super-sequence of α also satisfies C_M. A constraint C_A

Constraint		Anti-moto	Mono	Succ
Item	$C_{item}(\alpha) \equiv (\forall i : 1 \leq i \leq len(\alpha), \alpha[i]\theta V)(\theta \in \{\subseteq, \supseteq\})$	Yes	No	Yes
	$C_{item}(\alpha) \equiv (\forall i : 1 \leq i \leq len(\alpha), \alpha[i] \cap V \neq \varnothing)$	Yes	No	Yes
	$C_{item}(\alpha) \equiv (\exists i : 1 \leq i \leq len(\alpha), \alpha[i]\theta V)(\theta \in \{\subseteq, \supseteq\})$	Yes	No	Yes
	$C_{item}(\alpha) \equiv (\exists i : 1 \leq i \leq len(\alpha), \alpha[i] \cap V \neq \varnothing)$	Yes	No	Yes
Length	$len(\alpha) \leq l$	Yes	No	Yes
	$len(\alpha) \geq l$	No	Yes	Yes
Super-pattern	$C_{pat}(\alpha) \equiv (\exists \gamma \in P \text{ s.t. } \gamma \sqsubseteq \alpha)$	No	Yes	Yes
Simple aggregates	$max(\alpha) \leq v, min(\alpha) \geq v$	Yes	No	Yes
	$max(\alpha) \geq v, min(\alpha) \leq v$	No	Yes	Yes
	$sum(\alpha) \leq v)$(with non-negative values)	Yes	No	No
	$sum(\alpha) \geq v)$(with non-negative values)	No	Yes	No
Tough aggregate	**g_sum:** $sum(\alpha)\theta v, \theta \in \{\leq, \geq\}$ (with positive and negative values)	No	No	No
	average:$avg(\alpha)\theta v$	No	No	No
RE	Regular Expression	No	No	No
Duration	$Dur(\alpha) \leq \Delta t$	Yes	No	No
	$Dur(\alpha) \geq \Delta t$	No	Yes	No
Gap	$Gap(\alpha)\theta\Delta t(\theta \in \{\leq, \geq\})$	Yes	No	No

Table 4.5: Characterization of commonly used constraints (for detailed description about the constraints, refer to [58])

is **anti-monotonic** if a sequence α satisfying C_A implies that any non-empty subsequence of α also satisfies C_A. A constraint is succinct if it can be specified using a precise "formula". According to the "formula", one can generate all the patterns satisfying a succinct constraints. A new framework for mining sequential patterns with *prefix anti-monotone* constraints. A constraint C_{pa} is called **prefix anti-monotonic** if a sequence α satisfying C_{pa} implies every prefix of α also satisfies C_{pa}. A constraint C_{pm} is called **prefix monotonic** if a sequence α satisfying C_{pm} implies that every sequence having α as a prefix also satisfies C_{pm}. A constraint is called **prefix-monotone** if it is prefix anti-monotonic or prefix monotonic. The authors showed that most of the commonly used constraints (Table 4.5) are prefix-monotone. And all anti-monotonic or monotonic constraints are prefix-monotone, i.e., prefix-monotone property is weaker than anti-monotone and monotone properties.

An algorithm similar to PrefixSpan, called Prefix-Growth (Figure 4.11), is proposed to find sequential patterns with prefix-monotone constraints.

4.3.3 Mining Frequent Episodes

Another problem in mining sequential data is the discovery of frequent episodes in event sequences, proposed by Mannila *et al.* [45, 44, 46]. In this section, we follow the authors' definitions and introduce the problem and solutions of finding frequent episodes.

Given a set E of *event types*, an *event* is a pair (A, t), where $A \in E$

Algorithm Prefix-Growth
Input: A sequence database S, minimum support threshold δ, prefix-monotone constraint C
Output: The complete set of sequential patterns satisfying C
Method: call Prefix-Growth(<>, S)
Subroutine Prefix-Growth($\alpha, S|\alpha$)
Parameters: α: prefix; $S|_\alpha$: the α-projected database
Method:

1. Let l be the length of α. Scan $S|_\alpha$ once, find length-$(l+1)$ frequent prefix in $S|_\alpha$, and remove infrequent items and useless sequences;
2. for each length-$(l+1)$ frequent prefix α' potentially satisfying the constraint C do
   (a) if α' satisfying C, then output α' as a pattern;
   (b) for $S|_{\alpha'}$;
   (c) call Prefix-Growth($\alpha', S|_{\alpha'}$)

Figure 4.11: Prefix-Growth Algorithm

and t is the occurrence time associated with the event type A, represented by an integer. An *event sequence* **s** on E is a triple (s, T_s, T_e). where $s =< (A_1, t_1), (A_2, t_2), ..., (A_n, t_n) >$ (where $A_i \in E$ and $t_i \leq t_{i+1}$) is an ordered list of event. T_s and T_e are integers representing the starting time and the ending time, and $T_s \leq t_i < T_e$ for all $i = 1, 2, ..., n$. A *window* on an event sequence $s = (s, T_s, T_e)$ is an event sequence **w**=(w, t_s, t_e), where $t_s < T_e$ and $t_e > T_s$, and w consists of those pairs (A, t) from s where $t_s \leq t < t_e$. The time span $t_e - t_s$ is called the *width* of the window **w**, denoted $width(\mathbf{w})$. Given an event sequence **s** and an integer *win*, $W(\mathbf{s}, win)$ denotes the set of all windows **w** on **s**,such that $width(\mathbf{w}) = win$. Figure 4.12 shows an example of an event sequence $\mathbf{s} = (s, 29, 68)$, where $s =< (E, 31), (D, 32), (F, 33), (A, 35), ..., (D, 67) >$. A window starting at time 35 (shown in solid line) is (<(A,35), (B,37), (C,38), (E,39)>,35,40).

An *episode* is a partially ordered collection of events occurring together. Episodes can be described as directed acyclic graphs(DAGs). Figure 4.13 shows three different episodes. Episode 1 is a *serial episode*: it occurs in a

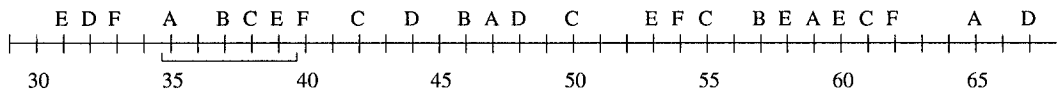


Figure 4.12: An example of event sequence and a window of width 5 [46]

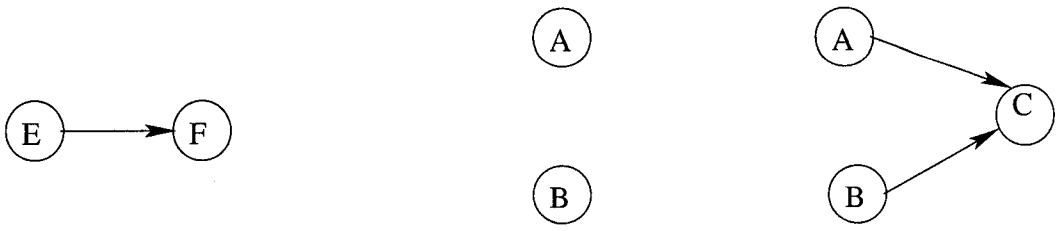
sequence only if there are event E and F that occur in that order in the sequence. Episode 2 is a *parallel episode*: there are no constraints on the order of event A and event B. Episode 3 is an example of non-serial and non-parallel episode: it occurs in a sequence if there are occurrences of A and B that precede an occurrence of event C, there are no constraints on the relative order of A and B. An episode is said to occur in a sequence if nodes of the episode have corresponding events in the sequence such that the event types are the same and the partial order of the episode is respected. For example, the window $(w, 35, 40)$ in Figure 4.12 contains events A, B, C and E. Episode 2 and 3 in Figure 4.13 occur in the window, but Episode 1 does not. An episode β is called a *subepisode* of α is the DAG for α is a subgraph of the DAG for β, e.g., Episode 2 is a subepisode of Episode 3 in Figure 4.13.

Given an event sequence $\mathbf{s}$ and a window width win, the frequency of an episode α in $\mathbf{s}$ is

$$fr(\alpha, \mathbf{s}, win) = \frac{|\{\mathbf{w} \in W(\mathbf{s}, win) | \alpha \text{ occurs in } \mathbf{w}\}|}{|W(\mathbf{s}, win)|}$$

Episode α is frequent if $fr(\alpha, \mathbf{s}, win)$ is no less than a threshold min_fr. The task of finding frequent episodes is to find all the frequent episodes from a given class ε of episodes. For example, the class could be all parallel episodes or all serial episodes. The collection of frequent episodes with respect to $\mathbf{s}$, win and min_fr is denoted by $F(\mathbf{s}, win, min\_fr)$.

Mannila *et al.* [45, 44, 46] proposed an algorithm called WINEPI for finding frequent episodes from event sequence based on the fact that *if an episode α is frequent in an event sequence $\mathbf{s}$, then all subepisodes $\beta \preceq \alpha$ are frequent.* Like GSP for finding frequent sequential patterns, WINEPI algorithm has two phases: candidate generation and candidate counting. But the difference is that WINEPI is designed for counting the number of occurrences of a pattern when moving a window along a single sequence, while GSP looks for patterns



Episode1 Episode2 Episode3

Figure 4.13: Examples of three different episodes represented by directed acyclic graphs

occurring in multiple sequences.

4.4 Summary

This chapter has introduced the work related to extracellular protein prediction. This problem falls into the general area of protein subcellular localization prediction. However, most existing methods for protein subcellular localization prediction aims at high overall accuracy, which is not appropriate in our case, since we are interested in identifying properties of amino acid sequences that discriminate extracellular proteins. Some approaches use additional information (e.g., lexical annotations) in addition to protein primary sequences. However, in our problem, we want to build our predictors only based on protein sequences. Also, we would like our predictors to provide as much interpretable justifications for biologists as possible.

Since the protein sequence is a kind of sequential data, we also surveyed various researches on mining sequential data, which have been studies extensively in data mining community. However, most algorithms introduced in this chapter cannot be easily applied in our case for the following reasons.

- In the sequential mining algorithms summarized in this chapter, a sequence is defined to be a list of elements, where each element consists of a set of items. Protein sequences can be modeled by this definition, by restricting each element to be a single item (amino acid). However, in

this case, most algorithms will find a lot of "uninteresting" patterns, in which a element only contain a single item. For example, a pattern "A->D->E->G->F" could be found by those algorithms. But this pattern is not quite interesting, since each element in this pattern only contains one item.

- The algorithms for finding episodes are used to find some "patterns" (episodes) in a single sequence, but we want to find those interesting subsequences from many extracellular proteins. Also, the structures of episodes have to be known beforehand. But in biology, we usually do not know the structures of subsequences that might be interesting.

The next chapter will elaborate on the methods that we have used for extracellular protein prediction.

Chapter 5

Extracellular Protein Prediction

This chapter introduces the features used for our prediction method, and the different data mining algorithm we use. Section 5.1 describes the features used in our algorithm and how to find those features from protein primary sequences. Sections 5.2, 5.3, 5.4 describe different data mining algorithms that we use for the prediction. Section 5.5 summarizes this chapter.

5.1 Feature Extraction

In this thesis, we use *frequent subsequences* as the features for the learning algorithms. A frequent subsequence is a subsequence made up of consecutive amino acids that occurs in more than a certain fraction (*MinSup*) of extracellular proteins. The reason we choose frequent subsequences is based on the following observations:

- Subsequences that appear frequently in extracellular proteins and rarely appear in intracellular proteins have very good discriminative power for identifying extracellular proteins. Those subsequences can be of great interest to biologists.
- It has been known that common subsequences among related proteins may perform similar functions via related biochemical mechanisms [23, 33, 39, 42].
- Frequent subsequences capture the local similarity that may relate to important functional or structural information of extracellular proteins.

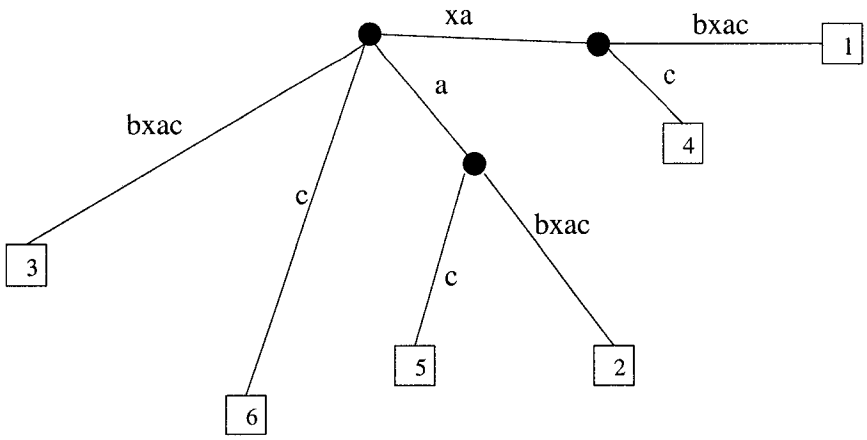


Figure 5.1: Suffix tree for string "xabxac" [28]

Frequent subsequences have been used for identifying outer membrane protein in Gram-negative bacteria proteins [69]. In this thesis, we want to use them for extracellular protein prediction.

In order to identify frequent subsequences from extracellular protein sequences, we use an efficient implementation of *generalized suffix tree* (GST) [80]. A suffix tree is trie-like data structure that compactly represent a string by collapsing a series of nodes having one child to a single node whose parent edge is associated with a string. A suffix tree for a m-character string S has exactly m leaves numbered from 1 to m. Each internal node has at least two children and each edge is labeled with an nonempty substring of S. No two edges out of a node have labels beginning with the same character. The concatenation of the edge labels on the root to leaf i (the numbers shown in rectangles in Figure 5.1) exactly corresponds to the suffix of S that starts at position i. Figure 5.1 shows an example of a suffix tree for the string "xabxac". The path from root to leaf numbered 1 spells out the string $S = xabxac$, while the path from the root to leaf numbered 4 spells out the suffix xac, which starts in position 4 of S. Suffix trees are used extensively in string matching. There are algorithms [73, 82] that can construct a suffix tree for a string in linear time.

A GST is an extension of the suffix tree, designed for representing a set of strings. Each suffix of the strings is represented by a leaf in the GST. Each

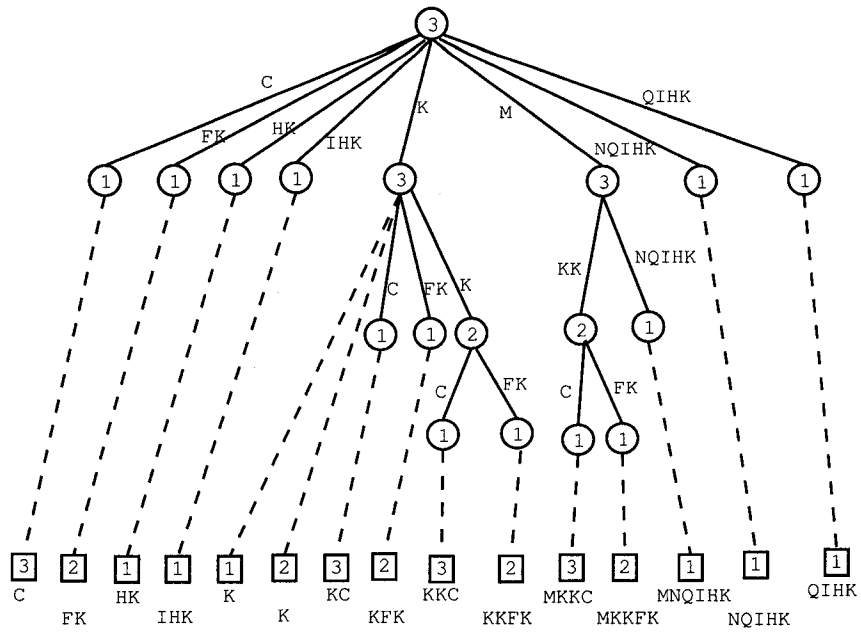


Figure 5.2: The GST for three sequences: MNQIHK, MKKFK and MKKC

leaf is associated with an index i. The edges are labeled with character strings such that the concatenation of the edge labels on the path from the root to the leaf with index i is a suffix of the ith string in the set. Figure 5.2 shows an example.

The algorithm for constructing the GST is as follows. A unique symbol is appended to the end of each sequence and all sequences are concatenated into a single one. The suffixes of the sequences are then inserted into a trie. When a node has only one child, the child is collapsed with its parent and the edge going down from the parent is labeled with a substring instead of a single character. In this way, a GST for all extracellular protein sequences are constructed [28].

In order to find the frequent subsequences, the GST is traversed to find all frequent subsequences that satisfy the minimum support threshold. In Figure 5.2, the number of occurrence of a subsequence is shown in a circle associated with an internal node. For example, starting from the root, if the path "K-K" is followed, we will reach an internal node with number 2 associated with it. This number tells us the occurrence of subsequence "KK" is 2.

5.2 Support Vector Machine

The support vector machine (SVM) approach for solving classification (especially binary classification) problems was proposed by Vapnik *et al.* [74, 75]. SVM is well founded theoretically because it is based on well developed statistical learning theory. It has also been shown to be very effective in real-world applications. For example, SVM has been successfully applied to handwritten digit recognition [16, 67], object recognition [63], text categorization [18], microarray data analysis [10], and protein secondary structure prediction [31]. Here, we use SVM for the problem of extracellular protein identification from primary sequences.

Here we briefly describe the basic idea behind SVMs. SVMs are usually used for binary (two-class) classification problems. SVMs assume data to be represented as vectors in some feature space. Assuming we have a set of training data, i.e., a set of input vectors $\vec{x}_i \in \mathbb{R}^d (i = 1, 2, ..., N, \mathbb{R} = setofrealnumbers)$ with corresponding labels $y_i \in \{+1, -1\} (i = 1, 2, ..., N)$, where d is the dimensions of the feature space and N is the number of training examples, $\{+1, -1\}$ are used to indicate the labels of two classes. In its simplest form, an SVM attempts to find a linear separator in the feature space that correctly separate the training data of two different classes. It seeks to maximize the *margin*, or separation between the two classes in order to improve the chance of accurate predictions of future data, as shown in Figure 5.3.

In most real applications, there are no linear hyperplanes that separate the original data in the original space. SVMs address this problem by mapping the input vectors $\vec{x}_i \in \mathbb{R}^d$ into a higher dimensional feature space $\Phi(\vec{x}) \in \mathbb{H}$. In this feature space, the linear separator is constructed. Figure 5.4 shows an example where non-separable data in two-dimensional input space is mapping into three-dimensional feature space, where a linear hyperplane can be found. The mechanism for this mapping $\Phi(\vec{x})$ is performed by a kernel function $K(\vec{x}, \vec{x}_i)$ which defines an inner product in the space $\mathbb{H}$. The space $\mathbb{H}$ is determined by the kernel function.

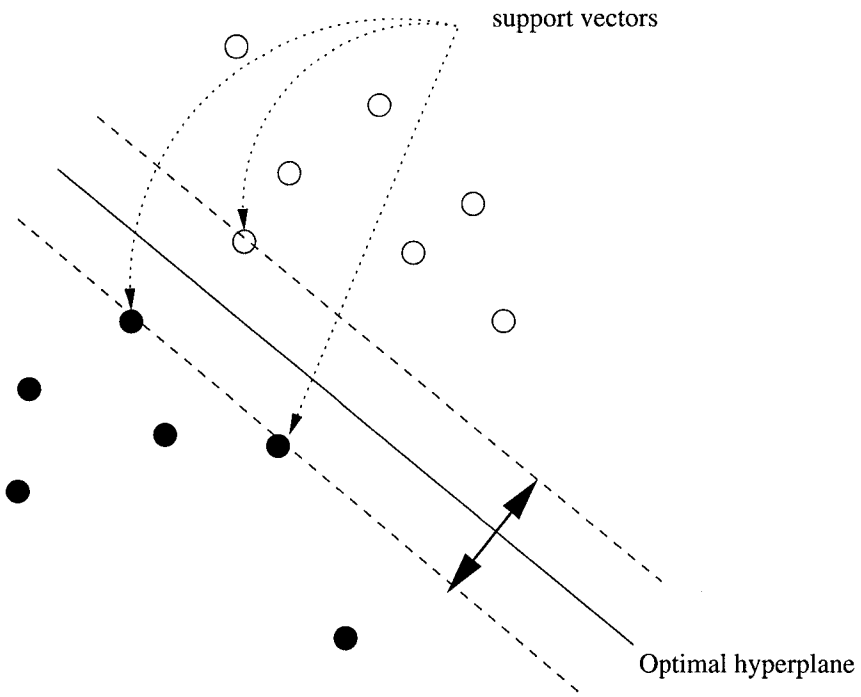


Figure 5.3: A linear SVM for a two-dimensional training set

The decision function implemented by SVM can be written as:

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

where α_i are obtaining by solving the following Quadratic Programming (QP) problem:

Maximize $\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \cdot y_i y_j \cdot K(\vec{x}_i, \vec{x}_j)$

subject to $0 \leq \alpha_i \leq C$

$\sum_{i=1}^{N} \alpha_i y_i = 0 (i = 1, 2, ..., N)$

where C is a regularization parameter that controls the trade off between margin and misclassification error. Those $\vec{x}_i$ are called *Support Vectors* if the corresponding $\alpha_i > 0$. There are efficient standard methods to solve the Quadratic Programming (QP) problem [15]. Due to the very high dimensionality of of the QP problem, an extension of the algorithm for solving QP is used in SVM algorithm [35].

Classical kernel functions include:

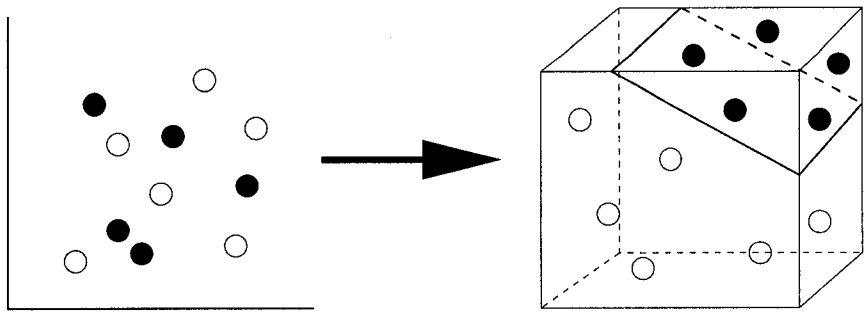


Figure 5.4: Support vector machine: mapping non-separable data from 2-D input space to 3-D feature space

- Linear Kernel Function:

$$K(\vec{x}_i, \vec{x}) = \vec{x}_i \cdot \vec{x}$$

- Polynomial Kernel Function:

$$K(\vec{x}_i, \vec{x}) = (\vec{x}_i \cdot \vec{x} + 1)^d$$

- Radial Basic Function(RBF):

$$K(\vec{x}_i, \vec{x}) = exp(-\gamma \parallel \vec{x}_i - \vec{x} \parallel^2)$$

To apply SVM approach in the extracellular protein prediction, the first step is to transform protein primary sequences, which are strings of letters, into some vector representation suitable for SVMs. Here, we use frequent subsequences as our feature space. These subsequences represent statistically discriminative features with regard to extracellular proteins. And the dimensionality is much lower than the feature space represented by all potential subsequences.

Each amino acid sequence is transformed into an n-dimensional vector $\vec{x} = (a_1, a_2, ..., a_n)$, where n is the number of frequent subsequences found from extracellular proteins, and $a_j (1 \leq j \leq n)$ is the feature corresponding to the ith subsequence. A binary representation is used. If the ith subsequence appears in protein sequence $\vec{x}$, the value of a_j is set to 1. Otherwise, it is

set to 0. For example, if there are 10 frequent subsequences, and a protein sequence contains only the first and the second frequent subsequences, the 10-dimensional vector will be (1,1,0,0,0,0,0,0,0,0). For the class label, +1 is used to indicate extracellular proteins and -1 for intracellular proteins.

SVM classifiers are trained in this transformed feature space with different kernel functions and different parameter settings. The results are described in Chapter 6.

5.3 Boosting

Boosting is a meta-learning method that has a theoretically justified ability to improve the performance of any *weak classifier*. A weak classifier is an algorithm that, given $\epsilon, \delta > 0$ and access to random examples, can achieve at least slightly better error rate ϵ than random guessing ($\epsilon > 1/2 - \gamma$, where $\gamma > 0$), with a probability $(1 - \delta)$. The purpose of boosting is to build a highly accurate classifier by combining many *weak* or *base* hypotheses, each of the weak hypothesis may be only moderately accurate. Various different boosting algorithms have been proposed in the literature [14, 22, 24, 64, 72].

Boosting algorithms work iteratively. During each iteration, a classifier is learned based on a different weighted distribution of the training examples. The main intuition behind boosting algorithms is to increase the weights of the incorrectly classified examples and decrease the weights of the correctly classified examples. This forces the learning algorithm to focus on those examples that are not correctly classified in the next iteration. The algorithm usually stops after a pre-specified number of iterations, or it can stop when some measurement of the quality of the classifier based on certain measurement (such as error rate) starts to deteriorate. The set of classifiers obtained after these iterations are combined together for the final prediction of unseen examples. Figure 5.5 shows the algorithm of the boosting algorithm.

In our application of extracellular protein prediction, we use AdaBoost [64, 24] with simple rule-based classifiers as the weak hypotheses. Every rule is a simple check for the presence or absence of a frequent subsequence in

Given: $(x_1, Y_1), (x_2, Y_2), ..., (x_m, Y_m)$ where $x_i \in \mathcal{X}, Y_i \subseteq \mathcal{Y}$
Initialize $D_1(i, l) = 1/(mk)$.
For $t = 1, 2, ..., T$:

- Pass distribution D_t to weak leaner.
- Get weak hypothesis $h_t : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_{t+1}(i, l) = \frac{D_t(i, l) exp(-\alpha_t Y_i[l] h_t(x_i, l))}{Z_t}$$

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

$$f(x, l) = \sum_{t=1}^{T} \alpha_t h_t(x, l).$$

Figure 5.5: AdaBoost Algorithm [65]

a protein primary sequence. Based only on the outcome of this test, the weak hypothesis outputs the prediction and the confidence that each label ("extracellular" or "intracellular") is associated with the protein sequence.

Formally, denote a subsequence by a, the possible class label for a protein sequence x by l (l could be "extracellular" or "intracellular"), and define $a \in x$ to represent the fact that subsequence a appears in protein sequence x. The weak hypothesis corresponding to this subsequence has the following form:

$$h(x, l) = \begin{cases} c_{0l} & \text{if a} \in \text{x} \\ c_{1l} & \text{if a} \notin \text{x} \end{cases}$$

where the c_{jl} are real numbers. For example, if subsequence a is "ABCDHIKG", l is "extracellular", $c_{0l} = 0.549$ and $c_{1l} = -0.168$. The weak hypothesis based on this subsequence would read in English:

IF subsequence "ABCDHIKG" appears in x
THEN predict
x to be "extracellular" with weight 0.549
ELSE predict

x to be "intracellular" with weight -0.168

The weak learner searches all possible frequent subsequences. For each subsequence, it calculates the value c_{jl} and assigns a score. Once all the subsequences are searched, the weak hypothesis with the lowest score is returned by the weak learner. In our case, the score will be an exact calculation of Z_t (refer to [64] for details). The score is calculated as follows [65]:

Let $X_0 = \{x : w \notin x\}$ and $X_1 = \{x : w \in x\}$. For $j \in \{0, 1\}$ and for $b \in \{-1, +1\}$, we calculate the following based on the current distribution D_t:

$$W_b^{jl} = \sum_{i=1}^{m} D_t(i, l)\{x_i \in X_j \wedge Y_i[l] = b\}$$

Z_t is minimized for a particular term by choosing

$$c_{jl} = \frac{1}{2} ln(\frac{W_{+1}^{jl}}{W_{-1}^{jl}})$$

and by setting α_t=1. These settings imply that

$$Z_t = 2 \sum_{j \in \{0,1\}} \sum_{l \in \mathcal{Y}} \sqrt{W_{+1}^{jl} W_{-1}^{jl}}$$

After all frequent subsequences are searched, the weak learner returns the one for which the value of Z_t is the smallest.

5.4 Frequent Subsequence Pattern (FSP) Method

In this section, we introduce another method for extracellular protein prediction, based on *frequent subsequence patterns* (FSP). The patterns we want to discover are regular expressions of the form $*X_1 * X_2 * ...$, where $X_1, X_2, ...$ are frequent subsequences made up of consecutive amino acids, and "*" is a variable-length-don't-care (VLDC) that can substitute for zero or more letters when matching the pattern against a protein sequence. Since we are interested in identifying extracellular proteins, subsequence patterns are mined only from extracellular proteins. We want to find those patterns that can discriminate extracellular proteins from intracellular ones. The reason we choose this form

of patterns is that subsequences capture local similarity that may relate to important structures or functions or extracellular proteins, and VLDCs compress the remaining irrelevant portions. An advantage of using subsequence pattern is because it can provide some biological insights for biologists. This form of patterns was proposed in [69] and has also been used to identify outer-membrane proteins in Gram-negative bacteria proteins.

Most sequential pattern mining algorithms are designed to find subsequence patterns in sequences of transactions. A protein sequence can be modeled by a sequence of transaction, where each transaction only contains one single item (amino acid). However most algorithms will find a lot of subsequence patterns that are not interesting. For example, "*A*C*E" is a possible pattern in sequence "(A)(B)(C)(D)(E)", but this pattern is not interesting to us, since "A" "C" "E" may not be frequent subsequences. And they can not find the patterns of the form "*ABC*DE", where "ABC" and "DE" are frequent subsequences. In [69], the authors use an exhaustive search to build patterns to identify outer membrane proteins by concatenating two or more frequent subsequences. However, since there could be thousands of subsequences found in the training set, exhaustive search produces an explosive number of candidate patterns. To deal with this problem, we exploit a greedy algorithm to find those patterns.

Our algorithm is based on the idea of FOIL (First Order Inductive Learner) proposed by Quinlan *et al.* [59]. FOIL repeatedly searches for the current best rule and removes all the positive examples covered by the rule until the positive examples in the data set are all covered. In our case, the positive examples are extracellular proteins and the negative examples are intracellular proteins. Figure 5.6 shows the algorithm FOIL.

The criteria that we use for choosing the best literal is based on its *Z-number* [38, 37]. Z-number is calculated as follows. Given a rule R and s_R denotes its support. Let a_C denote the mean of the target class C, defined as $a_C = |S_C|/|S|$, where S is the current training set and S_C is the subset of S where C is the class label. Let σ_C denote the standard deviation of the target class C. In the binary classification problem, it is calculated as

Algorithm FOIL
Input: Training set $D = P \cup N$. (P and N are the sets of positive and negative examples, respectively.)
Output: A set of rules for predicting class labels
Method:

```
1.  rule set R ← ∅
2.  while |P| > 0
3.    N' ← N, P' ← P
4.    rule r ← empty_rule
5.    while |N'| > 0 and r.length < max_rule_length
6.      Choose the best literal according to P' and N'
7.      append p to r
8.      remove from P' all examples not satisfying r
9.      remove from N' all examples not satisfying r
10.   end
11.   R ← R ∪ {r}
12.   remove from P all examples satisfying r's body
13. end
14. return R
```

Figure 5.6: FOIL Algorithm

$\sigma_C = \sqrt{a_C(1 - a_C)}$. Using these notions, Z-number is defined as

$$Z_R = \sqrt{s_R}(a_R - a_C)/\sigma_C$$

Z-number measures how well a rule R discriminate examples of class C. It is similar to the z-test or t-test in statistics. A rule with high positive Z-number predicts the presence of C with high confidence. A rule with high negative Z-number predicts absence of C with high confidence. A rule with Z-number close to zero does not has much power of discriminating examples of class C.

The FOIL algorithm is very efficient, but not very effective (in terms of prediction accuracy), due to the fact that whenever a rule is generated, all the positive examples covered by this rule will be removed, which results in the final rule set generated to be very small. In our case, since we want the rule in the format of "pattern of subsequences $\Rightarrow$ extracellular", a small rule set means the recall is very low, i.e., a lot of extracellular proteins in the test

set are not covered by the rule set, thus misclassified as intracellular proteins. In [85], the authors proposed an idea of decreasing the weight of an example after it is covered by a newly generated rule, instead of removing it. This will produce more rules and it is more likely to cover most of the positive examples in the test set. We adopt this idea in our frequent subsequence pattern method. However, there are some modifications made to the rule induction.

- The patterns we are interested in have the format of "$*X_1 * X_2 * ... \Rightarrow$ *extracellular*", i.e., the relative order of frequent subsequences (X_1, X_2,...) matters. Also it is possible for a pattern to contain multiple occurrences of a subsequence, i.e., "$*X_1 * X_2 * X_1*$" is a valid pattern.

- Consider the two examples in Figure 5.7. Pattern P1=$*X_1 * X_2*$ appears in protein sequence 1, pattern P2=$*X_3 * X_4*$ appears in sequence 2. Intuitively, P1 is more likely to be biologically significant than P2, since the two subsequences X_1 and X_2 are close to each other in P1 , while subsequences X_3 and X_4 are too far apart. In our algorithm, we introduce another parameter called $MaxGap$. When matching a pattern against a protein sequence, if the distance (in terms of number of amino acids) of two adjacent subsequences are too far apart, we do not consider it to be a match. For example, if $MaxGap$ is set to be 3, the pattern "*ABC*DEF*" does not match the sequence "ABCMNOPQDEF", since the gap between subsequence "ABC" and "DEF" is 5 (see Figure 5.8(a)). However the pattern "*ABC*DEF*" matches the sequence "ABCABCPQDEF", since we can find a way to align them, so that the gap between "ABC" and "DEF" is 2. We understand this is not a theoretically justified assumption in biological domain. But through manual inspection of our dataset and other publications [34], we decide to use it in our method.

The algorithm for finding patterns are shown in Figure 5.9. The procedure $Match(t, r, MaxGap)$ in Figure 5.9 is implemented by enumerating all the possible alignment of the subsequences in the pattern r to the sequence t.

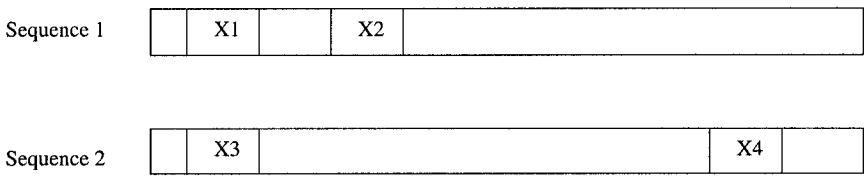


Figure 5.7: Examples showing two patterns

Figure 5.8: Matching pattern against sequence

The pattern r is considered to "match" sequence t, if there is one possible alignment, such that the distances between two adjacent subsequences are all less than $MaxGap$. For example, there are two possible alignments of the pattern "$*ABC*DEF*$" against the sequence "$ABCABCPQDEF$", shown in Figure 5.10. If $MaxGap$ is set to 3, we consider the pattern "$*ABC*DEF*$" matches sequence "$ABCABCPQDEF$", since in Figure 5.10(b), the alignment satisfies the $MaxGap$ constraint.

After the set of patterns are generated, we filter them in order to keep those patterns with good predictive power. Only those patterns with support greater than a threshold $MinSup$ and confidence greater than $MinConf$ are kept for predicting unseen protein examples.

The prediction process is relatively easy. Given an unseen example t, every pattern r in the pattern set is tested. If there exist a pattern r that matches t, t is predicted to be an extracellular protein, otherwise it is predicted to be an intracellular protein.

5.5 Summary

In this chapter, we have proposed several frequent-subsequence-based algorithms for building predictors of extracellular proteins. The reason for using frequent subsequences is that they capture local similarities that may relate

Algorithm FindPattern
Input: Training set $D = P \cup N$. (P and N are the sets of extracellular and intracellular proteins, respectively.)
Output: A set of patterns in the format of $*X_1 * X_2 * ...$ for predicting extracellular proteins
Parameters: α: rate of weight decreasing; δ: threshold for exiting the procedure; *min_Znumber*: minimum acceptable Z-number; *MaxGap*: maximum acceptable gap between two subsequences.
Method:

```
1.  set the weight of every example in P to 1
2.  pattern set R ← ∅
3.  totalWeight ← TotalWeight(P)
4.  while totalWeight > δ · totalWeight
5.     N' ← N, P' ← P
6.     pattern r ← empty_rule
7.     while true
8.        Choose the subsequence p with the largest Z-number, according to
N' and P'
9.        if Z_number(p) < min_Znumber then break
10.       append p to r
11.       for each example t in P' ∪ N'
12.          if not Match(t, r, MaxGap) then
13.             remove t from P' ∪ N'
14.          end
15.       end
16.    R ← R ∪ {r}
17.    for each example t in P
18.       if Match(t, r, MaxGap) then
19.          t.weight ← α · t.weight
20.       end
21.    end
22. return R
```

Figure 5.9: Algorithm for finding patterns

Figure 5.10: Two possible alignment of pattern "$*ABC * DEF*$" against sequence "$ABCABCPQDEF$"

to important functional or structural information of extracellular proteins, thus may have good discriminative power for identifying extracellular proteins. SVM and boosting are used for learning the classifier because they are well founded theoretically and have been shown to be very effective in many real-world applications. The frequent subsequence pattern method is used because the patterns found by this method can be easily interpreted by biologists and could be very useful for future biological analysis.

In the next chapter, we test the performance of our algorithms on a real dataset and compare them to other algorithms.

Chapter 6

Experiments

This chapter describes the experimental results of our methods, including SVM based on frequent sequences, boosting based on frequent subsequences, and the frequent subsequence pattern method. We compare our methods to SVM based on amino acid composition and boosting based on amino acid composition. We also investigate the effect of combining subsequences and amino acid composition, and the effect of removing some of those subsequences that are shorter than a certain threshold.

6.1 Dataset and Evaluation

Our predictor will eventually be used in extracytosolic plant proteins. However, since there are not that many proteins currently available in our database, we test the performance on a plant protein dataset that we got from the Proteome Analyst project [43] at the University of Alberta. This dataset was constructed from SWISS-PROT and contains 3293 proteins, among which 171 are extracellular proteins.

We performed 5-fold cross validation, i.e., each run takes one of the 5 folds (i.e., 1/5 of the data) as the test set and the remaining 4 folds (i.e., 4/5 of the data) as the training set. To ensure fair comparisons, all the methods are evaluated using the same folding. The number 5 was chosen arbitrarily based on other publications [43].

The performance of a classification algorithm is usually evaluated by its *overall accuracy*. However, in our application, overall accuracy is not a good

	Actual Extracellular	Actual Intracellular
Predicted as Extracellular	TP	FP
Predicted as Intracellular	FN	TN

Table 6.1: Confusion Matrix

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
2658	2605	2532	2817	2722

Table 6.2: Number of subsequence in each fold

evaluation metric. For example, in our dataset, only about 5% of the proteins are extracellular proteins. A high accuracy (95%) can easily be achieved by classifying every protein to be intracellular. Another possible choice for evaluation is to measure of prediction accuracies for extracellular and intracellular separately. But in this case, the performace of an algorithm is measure by two number (accuracy of extracellular proteins and accuracy of intracellular proteins). We cannot easily tell whether an algorithm is good, if it performs well on one measurement and poorly on another measurement, unless there is an appropriate way to combine these two measurements. In our experiment, we choose to use *F-measure* with respect to the rare class (extracellular proteins) as our evaluation metrics. They are based on the confusion matrix shown in Table 6.1. Using the notions in Table 6.1, precision (P) and recall (R) of extracellular class can be defined as:

$$P = \frac{TP}{TP + FP}, R = \frac{TP}{TP + FN}$$

F-measure (F) [37] is a metric that combines precision and recall. It is calculated as:

$$F = \frac{2PR}{P + R}$$

If a naïve classifier predicts every protein to be intracellular, the recall (R) of this classifier would be zero, resulting F-measure to be zero. So F-measure is a evaluation metric suitable in our case.

For all the experiments, the subsequences are obtained by setting the minimum support threshold to be 5%. The number of subsequences in each fold is summarized in Table 6.2.

		Recall	Precision	F-measure
Linear kernel	Default C	0.291	1	0.451
	C=10	0.693	0.957	**0.804**
	C=100	0.693	0.957	**0.804**
	C=1000	0.693	0.957	**0.804**
Polynomial Kernel (d=2)	Default C	0.26	1	0.413
	C=10	0.543	0.945	0.69
	C=100	0.543	0.945	0.69
	C=1000	0.543	0.945	0.69
RBF Kernel (γ=0.005)	Default C	0.11	1	0.199
	C=10	0.449	1	0.62
	C=100	0.449	1	0.62
	C=1000	0.449	1	0.62

Table 6.3: SVM classification with frequent subsequences

Number of iterations	Recall	Precision	F-measure
500	0.598	0.884	0.714
1000	0.606	0.917	**0.729**
2000	0.606	0.906	0.726

Table 6.4: AdaBoost classification with different number of iterations

6.2 Experiment Result of SVM

In this set of experiments, the SVM^{light} implementation [36] is used since it is well-known and has been used extensively in previous research. We tried with three different kernels, including the linear kernel, the polynomial kernel with degree of 2 and the radial basis function kernel with γ=0.005. For each kernel, we tried different values for C (the regularization parameter that controls the trade-off between margin and misclassification error). The result is summarized in Table 6.3. The best result (in terms of F-measure) obtained is 0.804 with a linear kernel.

6.3 Experiment Result of Boosting

In the experiments of boosting, we chose the number of iterations to be 500, 1000 and 2000. And we used BoosTexter [65] for the experiments. The results are shown in Table 6.4.

The results show that the boosting algorithm is robust with respect to the

Recall	Precision	F-measure
0.614	0.765	0.681

Table 6.5: Experiment Result of Subsequence Pattern Method

number of iterations. The best result obtained by boosting is 0.729 with 1000 iterations. The performance of boosting is comparable to that of SVM.

6.4 Experiment Result of the FSP Method

For the experiment of subsequence pattern method, there are quite a few parameters to be tuned. In order to tune those parameters, we took a portion of training examples and tried our algorithm on it with different parameter setting, then tested the learned model on another portion of the examples. Through extensive trial and error, we finally got the following parameter setting. $MinLen$ is set to be 3, min_gain to be 0.1, δ to be 0.03 and α to be 0.8. The $MinSup$ to be 5%, $MinConf$ to be 80%, $MaxGap$ to be 300. The result is shown in Table 6.5.

6.5 Comparison with Other Methods

We compared our methods with SubLoc [32]. SubLoc used SVM with amino acid compositions as its features. The authors show that SubLoc performs better compared with other methods based on amino acid composition. It also performs better than methods based on N-terminal signals. SubLoc is not specifically designed for predicting extracellular proteins, but since its implementation is based on SVM^{light}, we re-implemented it with SVM^{light} and tested it on our datasets. We tried the same parameter settings as we did in SVM with subsequences, the result is shown in Table 6.6. In the table, "nan" represents the non-valid number caused by 0 being the divisor. The best result obtained is 0.522 with polynomial kernel (d=2) and C=1000.

For the purpose of comparison, we also tried AdaBoost based on amino acid composition. Since the attributes are continuous values in this case, the weak hypothesis used is a single test of whether the composition of an amino

		Recall	Precision	F-measure
Linear Kernel	Default C	0.094	1	0.173
	C=10	0.055	1	0.104
	C=100	0.26	0.825	0.395
	C=1000	0.315	0.784	0.449
Polynomial Kernel (d=2)	Default C	0.094	1	0.173
	C=10	0.094	1	0.193
	C=100	0.3	0.844	0.442
	C=1000	0.378	0.842	**0.522**
RBF Kernel (γ=0.005)	Default C	0.094	1	0.173
	C=10	0	nan	nan
	C=100	0	nan	nan
	C=1000	0.063	1	0.119

Table 6.6: SVM classification with amino acid composition

Number of iterations	Recall	Precision	F-measure
500	0.48	0.678	0.562
1000	0.488	0.697	**0.574**
2000	0.472	0.652	0.548

Table 6.7: AdaBoost classification on amino acid composition

acid is above or below some threshold (see [65] for details). The result is shown in Table 6.7. The best result obtained is 0.574 with 1000 iterations.

For cross comparison, we choose the best (in terms of F-measure) result generated by each algorithm (i.e., 0.804 for SVM with subsequences, 0.729 for boosting with subsequences, 0.522 for SVM with amino acid composition, 0.574 for boosting with amino acid composition). The comparison of different algorithms is shown in Figure 6.2. Our methods based on frequent subsequences are better than methods based on amino acid composition. In particular, the SVM method based on frequent subsequences performs the best among different approaches.

Even though the SVM method based on frequent subsequences achieves the best experiment result, there are some advantages in using the FSP method. The reason is that the decision functions learned by SVM algorithms are difficult for people to understand. However, the decision rules found by the FSP method can be easily interpreted and modified by human experts. Figure 6.1 shows some examples of the rules found by subsequence pattern method. Biol-

IF (sequence contains *CKN*CGPGHGIS*) THEN (extracellular)
IF (sequence contains *YWGQNG*EIN*) THEN (extracellular)
IF (sequence contains *QVY*AGH*NVT*) THEN (extracellular)
...
ELSE (intracellular)

Figure 6.1: Examples of patterns found by the FSP method

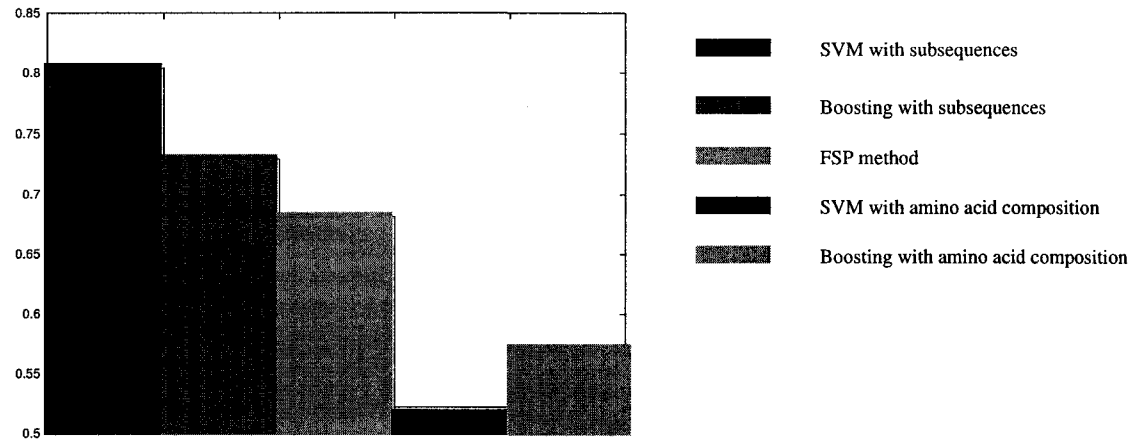


Figure 6.2: F-measures of different algorithms

ogists can easily read these rules and determine whether they are biologically meaningful. They can also incorporate their biological knowledge and modify the patterns, e.g., by adding or removing subsequences in the patterns, to get even better classification models.

6.6 Combining Frequent Subsequences and Amino Acid Composition

Since the methods based on frequent subsequences perform better than those based on amino acid composition, an interesting question is whether we can combine these two kinds of features to do a better job. We did some experiments to show the effect of combining them.

Table 6.8 shows the results of SVM based on combined features. Table 6.9 shows the comparison of F-measures of SVM based on combined features, frequent subsequences, and amino acid composition. The result shows that

		Recall	Precision	F-measure
Linear Kernel	Default C	0.291	1	0.451
	C=10	0.693	0.957	**0.804**
	C=100	0.693	0.957	**0.804**
	C=1000	0.693	0.957	**0.804**
Polynomial Kernel (d=2)	Default C	0.26	1	0.413
	C=10	0.543	0.945	0.69
	C=100	0.543	0.945	0.69
	C=1000	0.543	0.945	0.69
RBF Kernel (γ=0.005)	Default C	0	nan	nan
	C=10	0	nan	nan
	C=100	0	nan	nan
	C=1000	nan	0	nan

Table 6.8: SVM classification with combined features

		Combined Feature	Subsequence	Composition
Linear Kernel	Default C	0.451	0.451	0.173
	C=10	0.804	0.804	0.104
	C=100	0.804	0.804	0.395
	C=1000	0.804	0.804	0.449
Polynomial Kernel (d=2)	Default C	0.413	0.413	0.173
	C=10	0.69	0.69	0.193
	C=100	0.69	0.69	0.442
	C=1000	0.69	0.69	0.522
RBF Kernel (γ=0.005)	Default C	nan	0.199	0.173
	C=10	nan	0.62	nan
	C=100	nan	0.62	nan
	C=1000	nan	0.62	0.119

Table 6.9: Comparison of SVM based on different features

there is no obvious benefits of combined features for SVM. In the case of RBF kernel, SVM based on combined features is even worse than SVM simply based on frequent subsequences.

Table 6.10 shows the results of boosting based on combined features. Table 6.11 show the comparison of F-measures of boosting based on combined features, frequent subsequences, and amino acid composition. We can conclude that, contrary to SVM, the performance of boosting (measured by F-measure) can be improved significantly by combining frequent subsequences and amino acid composition.

Number of iterations	Recall	Precision	F-measure
500	0.685	0.967	0.802
1000	0.717	0.989	**0.831**
2000	0.708	0.989	0.826

Table 6.10: AdaBoost classification with combined features

Number of iterations	Combined feature	Subsequence	Composition
500	0.802	0.714	0.562
1000	0.831	0.729	**0.574**
2000	0.826	0.726	0.548

Table 6.11: Comparison of AdaBoost based on different features

6.7 Effects of *MinLen*

In SVM and Boosting, all the subsequences are used as features. However, short subsequences are more likely to appear than long subsequences. Some very short subsequences (e.g., subsequences with one amino acid or two) actually do not contain much information, even though they appear very frequently. In order to investigate the effect of the minimum length of subsequences, we did some experiments by removing those subsequences with length less than a certain threshold (*MinLen*). The results are shown in Table 6.12 and Table 6.13. We can see that in general, removing short subsequences does not improve the result.

		MinLen=1 (2667)	MinLen=2 (2647)	MinLen=3 (1851)	MinLen=4 (258)	MinLen=5 (67)
Linear Kernel	Default C	0.451	0.476	0.52	0.565	0.132
	C=10	0.804	0.804	0.67	0.5	0.111
	C=100	0.804	0.804	0.65	0.489	0.126
	C=1000	0.804	0.804	0.66	0.485	0.088
Polynomial Kernel (d=2)	Default C	0.413	0.413	0.392	0.517	0.565
	C=10	0.69	0.625	0.5	0.497	0.565
	C=100	0.69	0.625	0.5	0.497	0.565
	C=1000	0.69	0.625	0.5	0.497	0.565
RBF Kernel (γ=0.005)	Default C	0.199	nan	0.031	0.104	0.591
	C=10	0.62	nan	0.046	0.158	0.591
	C=100	0.62	nan	0.046	0.158	0.591
	C=1000	0.62	nan	0.046	0.158	0.591

Table 6.12: F-measure of SVM on frequent subsequences with different MinLen (the number in brackets shows the average number of frequent subsequences)

Number of iterations	MinLen=1 (2667)	MinLen=2 (2647)	MinLen=3 (1851)	MinLen=4 (258)	MinLen=5 (67)
500	0.714	0.704	0.613	0.556	0.551
1000	0.729	0.711	0.615	0.556	0.548
2000	0.726	0.724	0.599	0.55	0.556

Table 6.13: F-measure of AdaBoost on frequent subsequences with different MinLen (the number in brackets shows the average number of frequent subsequences)

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this dissertation, we have presented EPPdb, an on-line database that has been developed for proteomic analysis of extracytosolic plant proteins. Similar to other 2-D PAGE databases, EPPdb provides textual and graphical web interfaces that allow biologists to query and populate the database. EPPdb also provides an open API that allow other applications to access it as a Web service.

We also augmented EPPdb with a repository of tools that can be used in data analysis and mining tasks. In particular, we introduced a tool that we developed for extracellular protein prediction. We have proposed several frequent-subsequence-based algorithms for this task and compared them with existing methods, which are based on amino acid composition. The experiments show that our algorithms perform better than amino acid composition based methods. The best result is achieved by SVM classification based on frequent subsequences. However, even though the experimental results of SVM and boosting based on frequent subsequences are the best, there are some advantages in using the FSP method. The reason is that the decision functions learned by SVM and boosting are difficult for people to understand. However, the decision functions of the FSP method are easily readable rules, which can be easily understood by human experts.

7.2 Future Work

There are a number of directions for possible future research.

First of all, we only use the protein primary sequences for training the predictor of extracellular proteins. If additional properties of proteins (e.g., secondary structures, functions) are available, future research can take these characteristics into account to make a more accurate prediction.

Moreover, in the algorithms we have developed, we did not use biological knowledge about the characterizations of extracellular proteins. It would be beneficial if we could embed prior knowledge about extracellular proteins into our prediction system.

In addition, EPPdb is still in its early stage of development. As the project progresses, other interesting data analysis and mining tasks may be identified. Future research is needed to develop further tools to assist biologists.

Bibliography

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *In Proceedings of the 11th International Conference on Data Engineering*, pages 3–14, 1995.

[2] S.F. Altschul and W. Gish. Local alignment statistics. *Methods in Enzymology*, 266:460–480, 1996.

[3] M.A. Andrade, S.I. O'Donoghue, and B. Rost. Adaption of protein surfaces to subcellular location. *Journal of Molecular Biology*, 276:517–525, 1998.

[4] V.M. Anoop, U. Basu, M.T. McCammon, L. McAlister-Henn, and G.J. Taylor. Modulation of citrate metabolism alters aluminum tolerance in yeast and transgenic canola overexpressing a mitochondrial citrate synthase. *Plant Physiology*, 132:2205–2217, 1999.

[5] R.D. Appel, A. Bairoch, J.-C. Sanchez, J.R. Vargas, O. Golaz, C. Pasquali, and D.F. Hochstrasser. Federated two-dimensional electrophoresis database: a simple means of publishing two-dimensional electrophoresis data. *Electrophoresis*, 17(3):504–506, 1996.

[6] C. Apte, F. Damerau, and S. Weiss. Towards language independent automated learning of text categorization models. In *Proceedings of the 17th Annual ACM SIGIR Conference*, 1994.

[7] Rolf Apweiler, Amos Bairoch, and Cathy H. Wu. Protein sequence databases. *Current Opinion in Chemical Biology*, 8:76–80, 2004.

[8] U. Basu, A.G. Good, T. Aung, J.J. Slaski, A. Basu, K.G. Briggs, and G.J. Taylor. A 23kD, aluminum-binding, root exudate polypeptide cosegregates with the aluminum-resistant phenotype in *Triticum aestivum*. *Physiologia Plantarum*, 106:53–61, 1999.

[9] B. Boeckmann, A. Bairoch, R. Apweiler, M.-C. Blatter, A. Estreicher, E. Gasteiger, M. J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The Swiss-Prot protein knowledgebase and its supplement trembl in 2003. *Nucleic Acids Research*, 31:365–370, 2003.

[10] M.P.S. Brown, W.N. Grundy, D. Lin, N. Cristianini, C.W. Sugnet, T.S. Furey, M. Ares, and D. Haussler. Konwledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of National Academy of Science USA*, 97:262–267, 2000.

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

[12] K. C. Chou and D. W. Elrod. Protein subcellular location prediction. *Protein Engineering*, 12(2):107–118, 1999.

[13] M. Claros and P. Vincens. Computational method to predict mitochondrially imported proteins and their targeting sequences. *European Journal of Biochemistry*, 241:779–786, 1996.

[14] W. Cohen and Y. Singer. A simple, fast and effective rule learner. In *In Proceedings of Annual Conference of American Association for Artificial Intelligence*, pages 335–342, 1999.

[15] T. F. Coleman and Y. Li. A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables. *SIAM Journal on Optimization*, 6(4):1040–1058, 1996.

[16] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–293, 1995.

[17] Genetics Computer Croup(GCG). SPScan. Wisconsin Package Version 10.2.

[18] H. Drucker, D. Wu, and V. Vapnik. Support vector machines for spam categorization. *IEEE Transaction on Neural Networks*, 10:1048–1054, 1999.

[19] Frank Eisenhaber and Peer Bork. Wanted: subcellular localization of proteins based on sequence. *Trends in Cell Biology*, 8:169–170, 1998.

[20] Olof Emanuelsson, Sϕren Brunak, and Gunnar von Heijne. Chlorop, a neural network-based method for predicting chloroplast transit peptides and their cleavage sites. *Protein Science*, 8:978–984, 1999.

[21] Olof Emanuelsson, Henrik Nielsen, Sϕren Brunak, and Gunnar von Heijne. Predicting subcellular localization of proteins based on their N-terminal amino acid sequence. *Journal of Molecular Biology*, 300:1005–1016, 2000.

[22] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan. AdaCost: Misclassification cost-sensitive boosting. In *In Proceedings of 6th International Conference on Machine Learning*, Bled, Slovenia, 1999.

[23] K. A. Frenkel. The human genome project and informatics. *Communications of the ACM*, 34(11):41–51, 1991.

[24] Y. Freund and R. 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.

[25] Jennafer L. Gardy, Cory Spencer, Ke Wang, Martin Ester, Gabor E. Tusnady, Istvan Simon, Sujun Hua, Katalin deFays, Christophe Lambert, Kenta Nakai, and Fiona S.L. Brinkman. PSORT-B: improving protein subcelllular localization prediction for gram-negative bacteria. *Nucleic Acids Research*, 31(13):3613–3617, 2003.

[26] Minos N. Garofalakis, Rajeev Rastogi, and Kyuseok Shim. SPIRIT: Sequential pattern mining with regular expression constraints. In *Proceedings of the 25th VLDB Conference*, Edinburgh, Scotland, 1999.

[27] Y. Gavel, L. Nilsson, and G. von Heijne. Mitochondrial targeting sequences. why "non-amphiphilic" peptides may still be amphiphilic. *FEBS Letters*, 235:173–177, 1988.

[28] Dan Gusfield. *Algorithms on Strings, Trees and Sequences.* Cambridge University Press, 1997.

[29] C. Hoogland, V. Baujard, J.-C. Sanchez, D. F. Hochstrasser, and R. D. Appel. Make2ddb: A simple package to set up a two-dimensional electrophoresis database for the world wide web. *Electrophoresis*, 18:2755–2758, 1997.

[30] C. Hoogland, J.-C. Sanchez, L. Tonella, P.-A. Binz, A. Bairoch, D. F. Hochstrasser, and R. D. Appel. The 1999 SWISS-2DPAGE database update. *Nucleic Acids Research*, 28:286–288, 2000.

[31] Sujun Hua and Zhirong Sun. A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach. *Journal of Molecular Biology*, 2001.

[32] Sujun Hua and Zhirong Sun. Support vector machine approach for protein subcellular localization prediction. *Bioinformatics*, 17(8):721–728, 2001.

[33] Larry Hunter. *Artificial Intelligence and Molecular Biology.* AAAI Press, 1993.

[34] A. Icev, C. Ruiz, and E. Ryder. Distance-enhanced association rules for gene expression. In *Proceedings of the 3rd ACM SIGKDD Workshop on Data Mining in Bioinformatics*, Washington, DC, USA, 2003.

[35] T. Joachims. Making large-scale SVM learning practical. In *Advances in Kernel Methods-Support Vector Learning.* MIT Press, 1999.

[36] T. Joachims. *Learning to Classify Text Using Support Vector Machines.* Kluwer, 2002.

[37] M. V. Joshi. On evaluating performance of classifiers for rare classes. In *Proceedings of 2002 IEEE International Conference on Data Mining*, Maebashi City, Japan, 2002.

[38] M. V. Joshi, R. C. Agarwal, and V. Kumar. Mining needles in a haystack: Classifying rare classes via two-phase rule induction. In *Proceedings of ACM SIGMOD Conference*, pages 91–102, Santa Barbara, CA, 2001.

[39] N. Kamel, M. Delobel, T.G. Marr, R. Robbins, J. Thierry-Mieg, and A. Tsugita. Data and knowledge bases for genome mapping: What lies ahead? In *Panel Presentation in the 17th International Conference on Very Large Data Bases*, Barcelona, Spain, 1991.

[40] Benjamin M. Lewin. *Genes.* Oxford University Press, 1999.

[41] D.D. Lewis and M. Ringuette. Comparison of two learning algorithms for text categorization. In *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval(SDAIR'94)*, 1994.

[42] R.J. Lipton, T.G. Marr, and J.D. Welsh. Computational approaches to discovering semantics in molecular biology. In *Proceedings of the IEEE*, volume 77, pages 1056–1060, 1989.

[43] Zhiyong Lu. Predicting protein sub-cellular localization from homologs using machine learning algorithms. Master thesis, 2002. Department of Computing Science, University of Alberta.

[44] H. Mannila and H. Toivonen. Discovery generalized episodes using minimal occurances. In *In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining(KDD'96)*, pages 146–151, Portland, OR, 1996.

[45] H. Mannila, H. Toivonen, and A.I. Verkamo. Discovering frequent episodes in sequences. In *In Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining(KDD'95)*, pages 210–215, Montreal, Canada, 1995.

[46] Heikki Mannila, Hannu Toivonen, and A.Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, pages 259–289, 1997.

[47] B. Marchal. *Applied XML Solutions.* SAMS publishing, 2000.

[48] Afshin Nadershahi. Prediction of cell localization, 2002. http://www.micab.umn.edu/8006/litreviews/afshin.pdf.

[49] Rajesh Nair and Burkhard Rost. Inferring sub-cellular localization through automatic lexical analysis. In *In Proceedings of the tenth International Conference on Intelligent Syetems for Molecular Biology*, pages 78–86. Oxford University Press, 2002.

[50] K. Nakai. A knowledge base for predicting protein localization sites in eukaryotic cells. *Genomics*, 14:897–911, 1992.

[51] K. Nakai. Protein sorting signals and prediction of subcellular localization. *Advances in Protein Chemistry*, 54:277–344, 2000.

[52] H. Nakashima and K. Nishikawa. Discrimination of intracellular and extracellular proteins using amino acid composition and residue-pair frequencies. *Journal of Molecular Biology*, 238(1):54–61, 1994.

[53] Henrik Nielsen, Sϕren Brunak, and Gunnar von Heijne. Review: Machine learning approaches for the prediction of signal peptides and other protein sorting signals. *Protein Engineering*, 12(1):3–9, 1999.

[54] Henrik Nielsen, Jacob Engelbrecht, and Sϕren Brunak. A neural network method for identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *International Journal of Neural Systems*, 8:581–599, 1997.

[55] Henrik Nielsen, Jacob Engelbrecht, Sϕren Brunak, and Gunnar von Heijne. Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites. *Protein Engineering*, 10(1):1–6, 1997.

[56] Henrik Nielsen and Anders Krogh. Prediction of signal peptides and signal anchors by a hidden markov model. In J. Glasgow et al., editor, *Proceedings of Sixth International Conference on Intelligent Systems for Molecular Biology*, volume 1, pages 122–130. AAAI Press, 1998.

[57] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and MC. Hsu. PrefixSpan: Mining sequential patterns effeciently by prefix-projected pattern growth. In *In Proceedings of the 17th International Conference on Data Engineering*, pages 215–224, 2001.

[58] Jian Pei, Jiawei Han, and Wei Wang. Mining sequential patterns with constraints in large databases. In *Proceedings of the 11th International Conference on Information and Knowledge Management*, McLean, VA, 2002.

[59] J. R. Quinlan and R. M. Cameron-Jones. FOIL: A midterm report. In *Proceedings of 1993 European Conference on Machine Learning*, pages 3–20, Vienna, Austria, 2002.

[60] A. Reinhardt and T.Hubbard. Using neural networks for prediction of the subcellular location of proteins. *Nucleic Acids Research*, 26(9):2230–2236, 1998.

[61] C. Robinson and R.J. Ellis. Transport of proteins into chloroplasts, partial purification of a chloroplast protease involved in the processing of important precursor polypeptides. *European Journal of Biochemistry*, 142:337–342, 1984.

[62] D. Roise. Recognition and binding of mitochondrial presequences during the import of proteins into mitochondria. *Journal of Bioenergy and Biomembrane*, 29:19–27, 1997.

[63] D. Roobaert and M.M. Hulle. View-based 3D object recognition with support vector machines. In *Proceedings of the IEEE Neural Networks for Signal Processing Workshop*, pages 77–84, Totowa, NJ, 1999. IEEE Press.

[64] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[65] R. Schapire and Y. Singer. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2):135–168, 2000.

[66] G. Schatz and B. Dobberstein. Common principles of protein translocation across membranes. *Science*, 271:1519–1526, 1996.

[67] B. Scholkopf, C. Burges, and V. Vapnik. Extracting support data for a given task. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 252–257, Menlo Park, CA, 1995. AAAI Press.

[68] H. Schutze, D.A. Hull, and J.O. Pederson. A comparison of classifiers and document representation for the routing problem. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval(SIGIR'95)*, pages 229–237, 1995.

[69] R. She, F. Chen, K. Wang, M. Ester, J. L. Gardy, and F. S. L. Brinkman. Frequent-subsequence-based prediction of outer membrane proteins. In *Proceedings of ACM SIGKDD 2003 Conference on Knowledge Discovery and Data Mining*, Washington, DC, USA, 2003.

[70] J. Soll and R. Tien. Protein translocation into and across the chloroplastic envelope membranes. *Plant Molecular Biology*, 38:191–207, 1998.

[71] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *In Proceedings of the 5th International Conference on Extending Database Technology*, 1996.

[72] K. M. Ting. A comparative study of cost-sensitive boosting algorithms. In *In Proceedings of 17th International Conference on Machine Learning*, pages 983–990, Stanford University, CA, 2000.

[73] E. Ukkonen. On-line construction of suffix-trees. *Algorithmica*, 14:249–260, 1995.

[74] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.

[75] V. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

[76] G. von Heijne. Patterns of amino acids near signal sequence cleavage sites. *European Journal of Biolchemistry*, 133:17–21, 1983.

[77] G. von Heijne. Signal sequences: the limits of variation. *Journal of Molecular Biology*, 184:99–105, 1985.

[78] G. von Heijne, J. Steppuhn, and S.G. Hermann. Domain structure of mitochondrial and chloroplast targeting peptides. *European Journal of Biochemistry*, 180:535–545, 1989.

[79] M. Waltner and H. Weiner. Conversion of a nonprocessed mitochondrial precursor protein into one that is processed by the mitochondrial processing peptidase. *Journal of Biological Chemistry*, 271:21226–21230, 1996.

[80] J. Wang, G. Chirn, T. Marr, B. Shapiro, D. Shasha, and K. Zhang. Combinatorial pattern discovery for scientific data: Some preliminary results. In *SIGMOD-94*, Minnesota, USA, 1994.

[81] Yang Wang, Osmar R. Zaiane, Randy Goebel, Jennafer L. Shouthron, Urmila Basu, Randy M. Whittal, Julie L. Stephens, and Greg J. Taylor. Developing a database for proteomic analysis of extracytosolic plant proteins. In *Second International Workshop on Biological Data Management (BIDM'2004)*, 2004.

[82] P. Weiner. Linear pattern matching algorithms. In *Proceedings of the 14th IEEE Symposium on Switching and Automata Theory*, pages 1–11, 1973.

[83] Y. Yang and C.G. Chute. An application of least squares fit mapping to clinical classification. In *Proceedings of the Annual Symposium on Computer Applications in Medical Care*, pages 460–464, 1992.

[84] Y. Yang and J.P. Pederson. A comparative study of feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

[85] Xiaoxin Yin and Jiawei Han. CPAR: Classification based on predictive association rules. In *Proceedings of 2003 SIAM International Conference on Data Mining*, San Fransisco, CA, 2003.

[86] M.J. Zaki. Efficient enumeration of frequent sequences. In *Proceedings of the 7th International Conference on Information and Knowledge Management*, 1998.

[87] M.J. Zaki. SPADE: An efficient algorithm for mining frequent subsequences. *Machine Learning*, 40:31–60, 2001.

Appendix A

Glossary

amphiphilic : of or relating to a molecule having a polar, water-soluble group attached to a nonpolar, water-insoluble hydrocarbon chain.
2-D PAGE (2-DE) : two-dimensional polyacrylamide gel electrophoresis.

Amino acid composition : relative frequency of twenty amino acids in a protein sequence.

Boosting : an iterative learning algorithm that improves the performance of any weak classifer.

consensus sequence : a sequence of nucleotides or amino acids in common between regions of homology in different but related DNA or RNA or protein sequences.
Curated databases : a protein database monitored by biologists. The redundancy of curated databases is removed by compiling all the reports for a given protein sequence into a single entry.

EPPdb : Extracytosolic Plant Protein Database.

Extracellular : situated or occurring outside a cell or the cells of the body.

Extracytosolic : synonym to "extracellular".

Frequent subsequences : subsequences made up of consecutive amino acids that occur in more than a certain threshold of extracellular proteins.

Gram-negative : a particular kind of bacteria.

GST : Generalize Suffix Tree, a data structure designed for representing a set of strings.

Homologous : having the same evolutionary origin.

HTML : HyperText Markup Language.

Intracellular proteins : proteins that are localized within the cell.

Mahalanobis distance : a distance measurement between two N dimensional points scaled by the statistical variation in each component of the point.

MySQL : a popular Open Source SQL database management system.

NuSOAP : a toolkit providing a simple API for building Web Services using SOAP technology.

PHP : Hypertext Preprocessor, a widely-used open source general-purpose scripting language.

Protein subcellular localization : the cellular compartment where proteins are localized.

Sequence repository : a protein databases that make no efforts to provide a non-redundant collection of sequences.

SOAP : Simple Object Access Protocol.

SVM : Support Vector Machine, a learning algorithm for solving two-class classification problems.

Weak classifier : a classifier that performs better than random guessing.

Web Services : a mechanism of communicating between two remote systems.

WSDL : Web Services Description Language, an XML document used to describe Web Services.

XML : Extensible Markup Language, a universally agreed markup language primarily used for information exchange.

Appendix B

Sample Code

The following is a sample PHP code of accessing EPPdb using Web Services. It aims to retrieve all the protein entries that contain "trypsin" in the "DE" lines.

```
<?php
require_once('nusoap.php');
$soapclient=new soapclient(
            'http://www.cs.ualberta.ca/~wyang/php/soapserver.php');

$arr=array('DE');
$rows=$soapclient->call('query_by_keyword',
                        array('atts'=>$arr, 'keyword'=>'trypsin'));

echo "Query results:<br>";
echo "--------------------------------<br>";
foreach ($rows[0] as $row){
  foreach ($row as $element){
    echo $element."<br>\n";
  }
  echo "<br>\n";
}
?>
```

The services currently available are listed in Table B.1. More services will be added as this project proceeds.

Name	Description	Parameters
query_by_ac	Retrieve all the entries containing a specific word in the AC line	$name: the word to be search for in AC line
query_by_de	Retrieve all the entries containing a specific word in the DE line	$name: the word to be search for in DE line
query_by_keyword	Specify the attributes to be searched and several keywords, retrieve all the entries that contain all the keywords in those specified attributes	$atts: list of attributes (e.g., AC, DE, etc) $keyword: list of keywords separated by spaces

Table B.1: Web services in EPPdb