ABSTRACT

Title of dissertation: Improving the Performance and Precision of Bioinformatics Algorithms

Xue Wu, Doctor of Philosophy, 2008

Dissertation directed by: Professor Chau-Wen Tseng
Department of Computer Science

Recent advances in biotechnology have enabled scientists to generate and collect huge amounts of biological experimental data. Software tools for analyzing both genomic (DNA) and proteomic (protein) data with high speed and accuracy have thus become very important in modern biological research. This thesis presents several techniques for improving the performance and precision of bioinformatics algorithms used by biologists.

Improvements in both the speed and cost of automated DNA sequencers have allowed scientists to sequence the DNA of an increasing number of organisms. One way biologists can take advantage of this genomic DNA data is to use it in conjunction with expressed sequence tag (EST) and cDNA sequences to find genes and their splice sites. This thesis describes ESTmapper, a tool designed to use an eager write-only top-down (WOTD) suffix tree to efficiently align DNA sequences against known genomes. Experimental results show that ESTmapper can be much faster than previous techniques for aligning and clustering DNA sequences, and produces alignments of comparable or better quality.

Peptide identification by tandem mass spectrometry (MS/MS) is becoming the dominant high-throughput proteomics workflow for protein characterization in complex samples. Biologists currently rely on protein database search engines to identify peptides producing experimentally observed mass spectra. This thesis describes two approaches for improving peptide identification precision using statistical machine learning.

HMMatch (HMM MS/MS Match) is a hidden Markov model approach to spectral matching, in which many examples of a peptide fragmentation spectrum are summarized in a generative probabilistic model that captures the consensus and variation of each peak's intensity. Experimental results show that HMMatch can identify many peptides missed by traditional spectral matching and search engines.

PepArML (Peptide Identification Arbiter by Machine Learning) is a machine learning based framework for improving the precision of peptide identification. It uses classification algorithms to effectively utilize spectra features and scores from multiple search engines in a single model-free framework that can be trained in an unsupervised manner. Experimental results show that PepArML can improve the sensitivity of peptide identification for several synthetic protein mixtures compared with individual search engines.

Improving the Performance and Precision of Bioinformatic Algorithms

by

Xue Wu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2008

Advisory Committee:
Professor Chau-Wen Tseng, Chair/Co-Advisor
Professor Nathan Edwards, Co-Advisor
Professor Steven L. Salzberg
Professor Samir Khuller
Professor Mihai Pop
Professor Stephen Mount

Acknowledgments

During the years of my Ph.D. study, so many dramatic changes have happened in the school, in the outside world and in myself. I owe my gratitude to all the people who have made it possible for me to complete this journey.

First and foremost I'd like to thank my advisor, Professor Chau-Wen Tseng for giving me such an invaluable opportunity to work him on all kinds of challenging and extremely interesting projects over the past five years. He has always made himself available for help and advice. Most importantly, he demonstrated to me what scientific research should be like by not only his own words and action, by also his attitude toward research. It has been a pleasure to work with and learn from such an extraordinary individual.

I would also like to thank my co-advisor, Dr. Nathan Edwards. Without his extraordinary theoretical ideas and computational expertise, this thesis would have been a distant dream. I also owe great gratitude to Professor Steven Salzberg and Professor Stephen Mount, who spent time with me on my various projects and give me valuable suggestions. I also owe great thanks to Professor Samir Khuller and Professor Mihai Pop for spending their valuable time to serve on my committee.

I would also like to acknowledge help and support from the staff members from UMIACS and CS department. Without their help, I wouldn't be able to do all of the experiments in this thesis.

Lastly, and most importantly, I wish to thank my parents. They bore me, raised me, supported me, taught me and loved me. To them I dedicate this thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Since the discovery of the DNA double helix in 1953 by James Watson and Francis Crick, scientists have been trying to understand biology in molecular terms. Subsequently, the discovery of automated techniques to sequence DNA in the late 1970s and the fast development of protein chemistry in 1980s have mad studying organisms at the molecular level increasingly effective. Today, the continuous advances in molecular biology and corresponding experimental techniques have produced a huge amount of biological data generated by different types of automated techniques. As a result, computational tools have become essential in both genomics and proteomics for analyzing these biological data with high speed and accuracy.

In this thesis, three carefully designed algorithms are presented to demonstrate that computer scientists can contribute to the area of computational biology by finding ways to improve the performance and precision of bioinformatics algorithms commonly used by biologists, in ways that yield practical benefit for their biological research. These three algorithms were especially designed to demonstrate the ability of high performance computing and statistical machine learning in boosting both the performance and precision of bioinformatics algorithms.

The remainder of this chapter will begin by providing a brief introduction to genomics, proteomics and bioinformatics. In addition to providing basic background

knowledge for the rest of the thesis, we also present reasons why high performance computing and statistical machine learning techniques are helpful for solving molecular biology problems. Our motivation for designing the algorithms is provided at the end of the chapter.

1.1 Genomics

Genomics is the study of the entire genome of an organism. It includes both determining the entire set of DNA sequences of an organism and understanding an organism's genetic mapping. Since DNA sequence stores genetic information of all known living organisms, it is very important to understand its structure and function.

1.1.1 DNA Contains Basic Genetic Information

DNA is a macromolecule composed of a sequence of deoxyribonucleotides each containing a base, a sugar, and a phosphate group. Since DNA contains four different kinds of nucleotide bases (adenine (A), guanine (G), thymine (T) and cytosine (C)), DNA sequences may be written as strings constructed from an alphabet of four letters, namely A, C, G, and T.

DNA's three-dimensional structure is a double helix, which was discovered by James Watson and Francis Crick in 1953. In this structure, sugar and phosphate groups form the backbone and complementary bases (A-T, C-G) are connected by hydrogen bond.

DNA contains the basic genetic information that each organism needs to live and reproduce. The complete set of DNA that contains entire hereditary information of an organism is called the *genome* of the organism. The part of the genome that contains the hereditary information (producing proteins) are called *genes*. Human genome is believed to contain up to 20,000–25,000 genes (this estimate may change as human genome sequencing projects progress).

1.1.2 Genome Sequencing and Assembly

Since the genome contains such important information about organisms, obtaining the genome sequences of living organisms is widely considered to be the first step towards a deeper understanding of genetics and biology in general. Early DNA sequencing methods were slow and labor intensive. For instance, Maxim-Gilbert chemical sequencing involved gel electrophoresis and radioactive labeling [MG77].

Development of the Sanger (dideoxy termination) method greatly increased the ease of DNA sequencing [SNC77]. The method can be used to determine short (500–1000) nucleotides sequences off the end of a DNA fragment (called *reads*). Once the process was automated using capillary electrophoresis by companies such as Applied Biosystems, fast and inexpensive large-scale DNA sequencing become feasible and eventually led to sequencing of the genomes of many species.

Because DNA sequencers only produce many short DNA sequences, *assembly* techniques were developed to reconstruct the original DNA sequence by looking for overlaps between the short reads. The success of assembly tools enabled entire

genomes to be sequenced [ISF+04] using *shotgun sequencing*, where many random fragments from a long DNA sequence are sequenced and assembled to construct the original sequence [SCH+82].

1.1.3 Gene Structure and Genomic Annotation

Sequencing genomes is only one goal of genomic research. Once a genome has been sequenced, *genome annotation* seeks to label each part of the genome with its function. An important part of genome annotation is discovering what genes are present and where they are located in the genome. The structure of a gene is described in Figure 1.1.

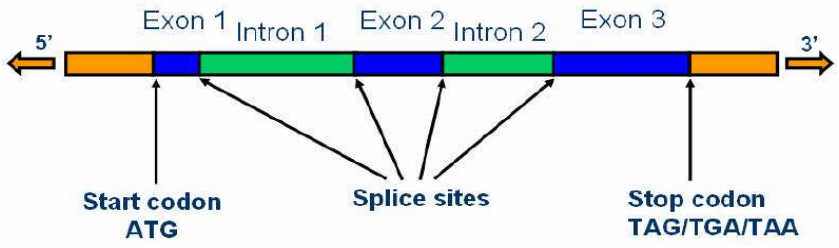

Figure 1.1: Gene Structure

As shown in the figure, genes are composed of sections of DNA known as exons and introns. Exons are regions that remain in mature mRNA and are later translated into proteins. Introns are regions that are *spliced* out of mRNA before translation to protein. The boundaries between introns and exons are called *splice sites*. Genomic annotation includes identification of the initiator of translation, splice sites, promoter and regulatory regions, and several other biologically important elements.

1.1.4 Gene Finding

Gene finding is the process of discovering genes in the genome. One problem with gene finding is caused by splicing. Since genes usually contain multiple introns, they may be produced from multiple regions of genomic DNA, depending on what introns have been spliced out. The issue is difficult because splice sites are not unique, and can occur in many different places in the genome.

Exacerbating the problem is *alternative splicing*, a process where the same genomic DNA can be spliced in different ways to produce multiple protein *isoforms*. In fact, alternative splicing is quite common, and makes determining the location of genes and correct splice sites even more difficult.

Experimental methods exist for accurate genome annotation (such as producing full-length cDNAs), but these methods are expensive, and too slow to keep up with the accelerating speed with which genomic sequence is being produced. For this reason computational methods play an essential part in gene finding.

Researchers have devised many algorithms for finding genes and their splice sites, some using statistic modeling techniques such as Hidden Markov Models. But since the types and location of splice sites are not completely known, the algorithms cannot locate all genes accurately using only statistic modeling.

Another computational approach to finding genes relies on a type of experimentally acquired data—short expressed DNA sequences called Expressed Sequence Tags (ESTs). ESTs are short (400-800 bases) single-pass sequences generated from expressed DNA and relatively easy to collect and sequence in the laboratory. How-

ever, despite their fragmentary and error-prone nature, ESTs are of interest because they can be collected from mature mRNA, and can thus be used as evidence for finding genes, their splice sites, and isoforms created from alternative splicing.

Because of the large size of genomes and enormous number of ESTs, developing algorithms to efficiently use EST information to find genes poses a computational challenge. Our first proposed algorithm targets mapping cDNA to genomes and demonstrates how to solve this problem with high performance computing techniques.

1.2 Proteomics

1.2.1 Proteins are Produced From DNA

Most of the properties of living organisms arise from the class of molecules known as proteins. Proteins are composed of hundreds to thousands of amino acid subunits connected in long chains. There are 20 different amino acids, so each can be represented by a single character in $[A, \ldots, Z]$. Because proteins are a product of DNA transcription and RNA translation as described by Figure 1.2, the order of amino acids is determined by genes encoded in the genome. Each amino acid chain folds in a different way to form a complicated 3 dimensional structure for each protein.

Proteomics is the study of all proteins of an organism. It strives to provide detailed information about protein structure, function and control of disease of biological systems. In the early stages of proteomics, protein chemistry was the major

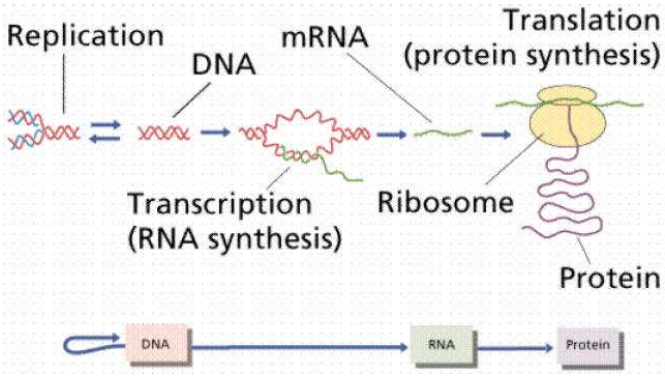

Figure 1.2: Protein Generation

research approach to studying protein and the link between gene and protein. In the 1980s and early 1990s, analytical protein chemistry mainly targeted improving the sensitivity of techniques for identifying proteins separated by gels. These protein sequencing results led to important information about proteins and their relationship with genes. At the same time, mass spectrometry became one of the major experimental techniques for analytical chemists to analyze small molecules. Advances in analytical protein chemistry, plus mass spectrometry and protein databases, have made large scale high throughput protein studies possible.

1.2.2 Major Disciplines in Proteomics

Currently there are three major disciplines in proteomics: mass spectrometry based proteomics, chip based proteomics, and genetic methods for studying the proteome. Among them, mass spectrometry based proteomics is a rapidly developing research area. Protein mass spectrometry is used for analyzing functional protein

complexes by identifying not only the members of the complexes, but also the interactions among the members. It is expected to become the main quantitative technology for systematical protein study.

However, because of the low quality of spectra data and high data generation speed, the analysis and interpretation of enormous amounts of protein mass spectra data is a major computational challenge. Even though computer scientists and statisticians have developed many algorithms and techniques for identifying proteins using experimental mass spectra, more precise and efficient algorithms are still needed.

1.3 Bioinformatics

1.3.1 Computational Processing of Biological Information

Defined as techniques for computational processing of biological information, bioinformatics requires knowledge from two areas: computer science and biology. Many fundamental biology problems from the early days of molecular biology have already demonstrated their need for computational solutions. Topics such as DNA structure, encoding of genetic information in protein, protein structure, biochemical pathways, etc., have motivated and defined the development agenda of bioinformatics and computational biology.

Bioinformatics as a research area started with combining computational and experimental information to better understand biological molecules. Construction of phylogenetic trees, understanding of DNA sequence properties and protein align-

ment properties were some early approaches for solving biology problems with computational algorithms. Later, with rapid development of automated experimental techniques, efficient computer algorithms were developed to cope with the fast growing amounts of biology data. Bioinformatics and computational biology became an independent discipline. Researchers in the area of bioinformatics are now striving to solve biology problems that pose great computational challenges.

1.3.2 Major Bioinformatics Research Topics

Some current bioinformatics research topics include:

- Pairwise and multiple sequence alignments
- Genome assembly
- Gene identification and annotation
- Gene expression analysis
- Protein sequencing
- Protein structure prediction
- Protein interaction
- Phylogenetic analysis

In general, bioinformatics algorithms strive to provide either the most accurate mathematical models for solving the biology problem, or techniques that can efficiently process large amount of experimental data in a reasonable period of time.

Bioinformaticians help biologists avoid or reduce expensive and lengthy wetlab experimental procedures.

In the remainder of the thesis, I will describe three algorithms to demonstrate the strength of using high performance computing and statistical modeling techniques in bioinformatics applications.

1.4 High Performance Computing

Two techniques widely used in the bioinformatics field are high performance computing and statistical machine learning. Both techniques play key roles in biological data analysis.

The necessity of high performance computing techniques arises from the advance of automated biological experimental techniques producing large data sets. In recent years, the amount of biological data coming from both genomics and proteomics have grown at a rapidly increasing rate. As a result bioinformatics researchers have started to take advantage of high performance computing architectures, including both distributed and shared memory platforms.

Many early high performance computing bioinformatics applications were in the area of pairwise sequence alignment. AGBLAST, NCBI BLAST [NCB], WU BLAST [WUB], mpiBLAST [DCF03], and UM-BLAST are different high performance versions of BLAST, a very popular sequence alignment tool. These versions of BLAST use parallel and distributed computer architectures to speed up the sequence alignment process. Later tools such as BLAST++ [WOOT03], megaBLAST,

TGICL, and PaCE exploit data properties and design new algorithms based on specific data structures to improve sequence alignment performance. Both approaches are necessary to improve the throughput and processing speed of bioinformatics tools such as BLAST.

More recently with increasing amounts and types of experimental data, high performance computing tools have been used in the realms of microarray gene expression data analysis, genetic networks, protein-protein interactions, phylogeny reconstruction, protein structure predictions, etc.

1.5 Statistical Machine Learning

Statistical machine learning is another key technique used in bioinformatics. It provides methods which make it possible to find patterns and automatically generate predictive models from large data sets, thus helping people understand and analyze the underlying biological information and mechanisms hidden in the data.

Machine learning plays a vital role in genomics research. Many machine learning algorithms have been developed to perform splice site prediction, alternative splicing form recognition, gene annotation and gene function prediction. In the area of proteomics, machine learning is used to aid protein structure and function prediction.

More recently, researchers have started to apply machine learning methods to peptide identification based on tandem mass spectrometry data. Machine learning has also been used to process microarray data to identify expression pattern and

study gene networks. In the area of system biology, machine learning has been used to model biological network to study genetic networks, signal transduction networks and metabolic pathways. In the area evolutionary biology, machine learning methods help reconstruct phylogenetic trees to study the evolution of many organisms. In addition, machine learning has also been used in primer design for PCR and to help analyze biological images.

Where high performance computing has increased the speed of bioinformatics algorithms, statistical machine learning has improved the accuracy of bioinformatics algorithms. Both classes of techniques are are very important tools for bioinformatics research.

1.6 Outline of Thesis

To demonstrate the application of both high performance computing techniques and machine learning techniques in bioinformatics research, two classes of algorithms are presented to solve two popular problems in genomics and proteomics fields.

The first class of algorithms are designed for aligning cDNA sequences to genomes. In genomics, the results of aligning cDNA sequences to genomes can be used to answer a number of important questions such as gene finding, EST clustering, finding alternative splicing isoforms, and identifying gene function. However, because of large genome sizes (thousands to millions of bases) and the huge number of available cDNA sequences (millions), aligning DNA to genomes can pose

a challenge for both accuracy and computation speed. In response we developed ESTmapper, an efficient cDNA to genome alignment algorithm that is carefully designed to align DNA to genomes with high accuracy and speed.

The second class of algorithms are designed to identify peptides by analyzing tandem mass spectra from protein samples. Tandem mass spectrometry is a key technique used for high-throughput protein identification. But due to limitations of current database search based protein identification algorithms, it is not easy to improve the accuracy of the results.

We present the design of two algorithms (HMMatch and PepArML) for improving peptide identification accuracy. HMMatch recognizes previously observed peptide fragmentation patterns to complement database search based methods. PepArML enhances database search by distinguishing between true and false positive protein identifications by combining multiple search engine results and exploring a large feature space.

The final part the thesis summaries the proposed algorithms and concludes with possible directions for future research.

Chapter 2

High-throughput EST/cDNA to Genome Mapping

2.1 Background

Recent advances in molecular biology techniques such as automated DNA sequencing have allowed scientists to quickly gather huge amounts of DNA sequence data. Two types of DNA sequence data are particularly interesting: genomic DNA and *expressed sequence tags* (ESTs). Genomes are long (thousands to millions of bases) DNA sequences representing the complete DNA of an organism, carefully constructed with high accuracy from many experiments. In comparison, ESTs are short (400-800 bases) single-pass DNA sequences that can be collected from mature mRNA that have already been spliced (introns excised). ESTs are relatively easy to collect and sequence in the laboratory but are more error-prone since each EST represents a single read.

With the increasing number of species whose genomes have been sequenced and the growing collection of EST sequence libraries (the August 2008 version of dbEST contained 54 million EST sequences from 1605 species), efficiently and accurately mapping large numbers of ESTs and other DNA sequences to genomes has become a challenging problem, but one of increasing importance to biologists. Such mappings may be used to answer a number of important questions, such as:

Gene finding. Determining which portions of the genome are actual genes is quite difficult. Many computational gene finding techniques exist for classifying and predicting genes based on the genomic DNA sequence alone. However, the accuracy of these techniques can generally be improved by including ESTs from mature mRNAs. To use this information, ESTs need to be mapped to the appropriate locations in the genome for inclusion gene finding [SKG04].

EST clustering. Algorithms also exist for finding genes based on ESTs alone, by forming clusters of overlapping ESTs and assemble the sequences in each cluster [BDH99, MCJ03, PGB02, LHP+00, PHL+03, KAKB03]. EST clusters may be formed by comparing ESTs against each other to find similarity and overlaps. Alternatively, clusters may also be formed from ESTs mapped to overlapping and nearby locations in the genome [CRL+04, LB04].

Alternative splicing. A major source of complexity for gene finding is the fact that genes can consist of non-contiguous sections of genomic DNA formed through splicing [DL99, SCK04]. Even worse, variations in splicing can cause a gene to produce multiple transcripts. Mapping ESTs to the genome can help point out splice sites and predict alternative splicing [GMXL04, CHV02].

Gene regulation. Whether genomic DNA is actually expressed depends on a number of factors, but it is known that gene regulation can be affected by DNA sequences near to but not actually part of the gene. Mapping ESTs and genes to the genome thus allows biologists to examine the DNA surrounding each gene to look for factors

affecting gene regulation.

Gene function. Using a number of unique markers in the genome, biologists have calculated linkage maps predicting the location in the genome of genes responsible for a number of genetic traits. Mapping genes to a location in the genome thus improves the ability of biologists to predict gene function to aid in disease diagnosis and drug design [PSL+04].

2.2 Related Work

Since the ability to align DNA sequences to the genome is so useful, researchers have investigated many techniques for performing such alignments. One way is to use traditional pairwise sequence alignment method.

2.2.1 Pairwise Sequence Alignment

Pairwise sequence alignment compares two sequences to find and align the most similar substrings based on some metric. It is probably the most commonly performed computation in bioinformatics, and many algorithms have been developed. Basic Local Alignment Search Tool (BLAST) [AGM+90] is the most popular and widely used tool for sequence alignment and similarity search. The search strategy is based on using scoring matrices to compare short subsequences (words) in the query sequence against the entire target DNA or protein sequence database to find statistically significant matches, then attempting to extend these matches to find the most similar sequences or subsequences.

BLAST speeds up local sequence alignment in 3 algorithmic steps. First, it compiles a list of words of length w. For proteins, the list only consists those words that score at least T when compared to some word in the query sequence. Next, BLAST scans through every database sequence to find all the occurrences of the words in the list. In the third step, the matching words are extended into ungapped local alignments between the query sequence and the sequence from the database. Extensions are continued until the score of the alignment drops below a threshold. The top-scoring alignments are combined into local alignments.

Researchers also designed high performance BLAST algorithms to improve its throughput when processing large sequence databases. Threaded BLAST [NCB, WUB], mpiBLAST [DCF03] and BLAST++ [WOOT03] represent three typical kinds of methods that are used to speed up the BLAST searching further. However, due to the large size of many genomes (around 3 billion bases for Human and Mouse) and the large number of DNA sequences collected by biologists, aligning DNA to genomes pose a computational challenge. Even using high performance BLASTs can be too expensive when a single high-throughput automated DNA sequencer (e.g., ABI Prism 3730xl) can output two million bases of sequence per day.

Producing accurate DNA to genome alignments is also problematic, since genomes contain many very similar if not duplicate DNA sequences that can result in multiple plausible alignments to different portions of the genome. Careful analysis is needed to distinguish between the possible alignments and calculate the most plausible mapping.

A second (somewhat subtle) issue can also reduce the effectiveness of pairwise

alignment tools when used to align DNA to genomes. Tools such as BLAST are typically designed and tuned to find homology between nearby sequences, for instance the same gene in Human and Mouse. As a result scores and statistical significance of matches (for instance, BLAST's substitution/scoring matrix) is usually calculated based on the similarity of two sequences after an evolutionary period.

In comparison, for DNA to genome alignments we can usually assume the DNA actually came from the genome, so the sequences should be highly similar. Any differences arise from splicing and sequencing errors, not as a process of mutation and evolution. As a result, we expect to find much longer identical common substrings when comparing query sequences to genomes when compared to typical BLAST searches. Techniques that take advantage of these observations may be able to improve the accuracy of their alignments.

2.2.2 cDNA to Genome Alignment

As we have thus seen, standard pairwise sequence alignment techniques face obstacles when applied to cDNA to genome alignment. As a result researchers have designed several other alignment techniques ([FHZ+98, Ken02, OM02, WW05, WCO01]).

megaBLAST. MegaBLAST is a program from the NCBI BLAST software suite that uses a greedy algorithm to align nucleotide sequences [ZSWM00]. The program provides good performance for highly similar sequences with minor differences, and is thus frequently used for genome alignments.

BLAT. BLAT is an alternative pairwise sequence alignment algorithm [Ken02]. BLAT maintains a precomputed hash table index of the locations of all non-overlapping substrings (words) of length k. Performance is dramatically improved because queries do not need to scan the entire sequence database. Only the sections of the sequence database with hits in the index need to be examined to compute more detailed local alignments and scores. Substrings that yield too many (hundreds or more) hits to the genome can be filtered out and ignored, or alignment time may increase dramatically.

Sim4. Sim4 is one of the oldest and most frequently used programs for aligning spliced DNA sequence with genomic sequence, allowing introns and a small number of sequencing errors [FHZ+98]. Unlike BLAST, it attempts to recognize biological valid splice sites for non-contiguous alignments to the genome. Sim4 was created because of the inefficiency of BLAST when mapping large numbers of cDNA sequences to genomes. Researchers use Sim4 for studying gene-to-genome annotation and alternative splicing. However, Sim4 can be slow since it uses dynamic programming. Older EST alignment tools such as *est_genome* [Mot97] and *est2gen* [GMP96] also use dynamic programming, and too slow to search entire genomes.

Spidey. Spidey is a computer program to align spliced sequences to genomic sequences [WCO01]. It is incorporated in the NCBI Toolkit for biologists to study gene annotation and alternative splicing. To find good alignments, Spidey uses NCBI BLAST to produce a list of candidate alignments, then refines the alignments

while considering splice sites.

Squall. Squall is a tool similar to BLAT, but especially designed to map ESTs to genomes [OM02]. Squall uses lookup tables to quickly find candidate substrings (21 bases) within 300 bases of the beginning and end of each EST sequence. It improves efficiency by discarding all candidates that map to more than ten locations in the genome. If the distance between the start and stop candidates is less than 3 million bases, a more precise algorithm is used to calculate and score possible EST to genome alignments.

The authors claim Squall is 100 times faster than BLAT. However, we find BLAT to be much faster than reported in their paper, indicating the authors may have not filter out excessively common substrings from the BLAT hash index as recommended. The authors report 0.03, 1.69, and 12 seconds to align each RefSeq sequence to human chromosome 22 using Squall, BLAT, and sim4 on a PrimePower 1000. In comparison, for the same data set on a SunFire 6800 we found it takes 0.02, 0.054, and 31 seconds using ESTmapper, BLAT, and sim4.

GMAP. GMAP is a recently developed tool for mapping and aligning cDNA sequences to genomes ([WW05]). It achieves high precision and efficiency by using a minimal sampling strategy for genomic mapping, starting with an index of 24-mers (stored as a pair of 12-mers). Matching oligomers between cDNA and the genome are chained together for approximate alignment, then dynamic programming (DP) is performed from both ends of the cDNA (sandwich DP) for splice site detection. In

addition, GMAP can attempt to identify microexons [VHS03] using statistical significance testing. GMAP also requires less memory by keeping a very small lookup table in memory and only reading in portions of the genome with high probability of alignments. In experiments GMAP was able to identify splice sites more accurately than BLAT, sim4, Spidey, and GeneSeqer for sets of human and Arabidopsis cDNAs. GMAP also proved to be highly efficient, providing about a 6-fold speed improvement over BLAT.

ESTmapper/sim4. We recently discovered another sequence alignment software also called ESTmapper which we will refer to here as ESTmapper/sim4. It was developed by Florea and Walenz [FW] and is referenced elsewhere [ISF+04, FFM+05]. ESTmapper/sim4 is also a software package for aligning cDNA onto genomic sequences. It uses a k-mer based algorithm to pre-compute indices for each genome, while we use suffix trees to pre-compute indices for each genome. ESTmapper/sim4 also couples with sim4 for generating spliced alignment results.

2.2.3 EST Clustering

Clustering is usually the first step in using ESTs for gene finding. Historically clustering algorithms compare ESTs against each other to form clusters. NCBI also maintains UniGene, a reference list of EST clusters automatically generated from ESTs in dbEST [WBB+03].

TGICL. TGICL (TIGR Gene Indices CLustering tools) is an example of a popular software system for clustering large EST data sets using pairwise comparisons [PHL+03]. TGICL uses mgBLAST, a modified version of megaBLAST that provides additional output filtering and uses a dynamic offset within a database for incremental searches. MgBLAST is used to quickly perform an all-to-all pairwise comparisons between EST sequences. Processing can be performed in parallel by partitioning the database, then merging compressed sorted files.

Clustering uses a greedy algorithm based on the best alignments, and known full-length cDNAs can be used as seeds to improve efficiency and produce larger clusters for incremental updates. Clusters output are passed to the CAP3 assembly tool [HM99] as multi-FASTA files and then assembled into high-quality consensus sequences. TGICL is used to generate the TIGR Gene Indices for 60 different species with between 10 thousand and 4 million EST sequences [LHP+00]. TGICL has been parallelized for PC clusters using PVM.

PaCE. PaCE (Parallel Clustering of ESTs) is one of the first clustering tools designed to exploit the power of suffix trees to avoid the need for all-to-all pairwise comparisons [KAKB03]. It first constructs (in parallel) a generalized suffix tree consisting of all EST sequences by first sorting ESTs and sending each to the appropriate processor. Once the suffix tree is complete, a variation of the algorithm for finding longest repeated substrings can be used to find all pairs of ESTs with common substrings above a certain threshold. The clustering algorithm then starts with pairwise comparisons between ESTs with long common substrings, greatly

increasing the likelihood of forming a cluster.

By using an on-demand algorithm for generating promising EST pairs, the PaCE approach generally requires only $O(n)$ number of pairwise comparisons, though $O(n^2)$ are still needed in the worst case. Additional refinements are needed to create and maintain clusters in parallel. Clusters produced by PaCE are of high quality when compared to a benchmark EST set from Arabidopsis created by aligning ESTs directly to the genome. A weakness of the PaCE system is that building a suffix tree for the input EST data set requires a large amount of memory, proportional to the number and size of ESTs. The authors were able ameliorate this problem by parallelizing their algorithm to run on a PC cluster, splitting the suffix tree so that each node only needs to hold a portion of the suffix tree in memory.

2.2.4 WOTD Suffix Tree

A suffix tree is a data structure that allows many problems on strings (sequences of characters) to be solved quickly and efficiently [Gus77]. It is formed by calculating and storing all the suffixes of a string in a *trie* structure. Suffix trees may be also *compressed* and *compacted*, storing entire substrings as indices to the original string to reduce storage.

Suffix trees are very efficient data structures. They can be constructed in $O(n)$ time, and finding longest common substrings between two sequences and longest repeated substrings only require $O(n)$ time. Unfortunately, there is a large (30+) multiplicative factor in the size of the suffix tree relative to the original sequence.

Researchers have investigated different efficient implementations and algorithms on suffix trees. One recently developed variation is called a *write-only, top-down* (WOTD) suffix tree [GJS03]. WOTD suffix trees are more expensive to build in that they require $O(nlog(n))$ expected and $(O(n^2))$ worst time to construct, but they require only a 10 to 12-fold increase in memory relative to the original sequence, and display good cache locality in performing searches.

WOTD suffix tree implementations may be *eager* (build the full WOTD tree immediately) or *lazy* (build portions of the WOTD tree as needed as queries are processed). In our research, we used the eager version of the WOTD implementation to avoid the expensive tree building overhead by building suffix trees once and using them whenever we need to align cDNA sequences to genomes. For *eager* implementation, the suffix tree is built starting from the root. The sub-trees are built recursively following the order of from top to bottom. The built suffix tree are stored in two flat arrays: root's children table and suffix tree table. This property of WOTD suffix tree makes it easy to build tree once and store the tree on the disk to use later.

2.3 ESTmapper: Efficiently Aligning DNA Sequences to Genomes

The traditional hash table based algorithms (BLAST [BPC+99], Spidey [WCO01], BLAT [Ken02], Sim4 [FHZ+98], etc.) usually return too many hits in the word searching step, thus causing the algorithms to spend a long time selecting and stitching together hits in the following steps. To speed up cDNA to genome alignment,

we designed ESTmapper [WLT05], our algorithm for aligning DNA sequences to genomes based on WOTD suffix tree for indexing the genomes.

The principle behind ESTmapper algorithm is still based on local sequence similarity. The assumption is that DNA sequences which can be mapped back onto genome should share a high degree of local sequence similarity with some specific area of genomic sequence, even considering introns and sequencing errors. The advantage of ESTmapper (in addition to fast substring search speed) is that the algorithm can filter out most short common substrings shared between DNA sequence and genome sequence, and only deal with a very small number of long common substrings that lead to high quality alignments. As a result, ESTmapper can process the sequences much faster than other algorithms, but hopefully without sacrificing the precision of the results.

2.3.1 Algorithm

The algorithm used by ESTmapper consists of the following steps:

Preprocess the genome. The genome sequence is read from a file and converted into a eager WOTD suffix tree. Preprocessing the genome only needs to take place once per genome, since the eager WOTD suffix tree can be stored as two flat arrays (root's children table and suffix tree table). The root's children table has a fixed maximum size, and suffix tree table has variable size. So after the suffix tree is built for each genome, the root's children table, suffix tree table size and the actual suffix tree table are written out to disk as a single file. When ESTmapper needs to use

the genome suffix tree, the two tables can be quickly read back into memory in the order they are stored on the disk.

Map DNA sequences to the genome. Each DNA sequence is mapped using the suffix trees for the genome, considering both its minus and plus strand. Mappings are computed as follows:

1. Find long common substrings

   Suffixes of each EST sequence are compared to the suffix tree for the genome to find long common substrings. Performance is improved in several ways. First, we discard common substrings that are below a user-specified minimal length, or that appear too often in the genome according to a user-specified frequency threshold. This step limits the number of random matches for the same common substrings, and reduces the need to examine common substrings that probably lead to low-quality alignments.

   Second, we avoid checking all suffixes of each EST sequence, by skipping past suffixes where possible. Skipping means the next long common string search will start from the current starting matching position plus a skip offset. The WOTD suffix tree does not maintain suffix links in order to reduce storage, so we do not know the ideal skip offset. Instead, we choose either the length of the current common substring on the query EST sequence, or a user-specified skip offset, whichever is greater. Since we extend common substrings in the next step, we do not need to find the longest common substring, just one long enough to be extended.

When the search is completed, the long common substring lengths and locations on both DNA and genome sequence are stored for next step of processing. We expect such long common substrings to be common, since the query EST sequence was most likely expressed by the genome in the first place.

2. Extend long common substrings

   Each long common substring found in the first step is extended in both directions. This substring extension step is similar to NCBI BLASTN's match extension step except ESTmapper's extension is based on long common substrings instead of matched k-mers. Then the extended substrings are sorted in order of their locations.

3. Build (spliced/gapped) mappings

   To handle splicing and gaps, the ESTmapper algorithm examines the list of long substrings and locations, and combines them into a single spliced/gapped *mapping region* if two substrings' mapping location on genome are sufficiently close to each other.

4. Refine the mappings/alignments

   ESTmapper looks through the mappings' component substrings to determine whether two substrings are close enough on both DNA and genome sequence that they can be merged into one with small gaps and/or mismatches in between. After the check, each substring should correspondent to one exon. Boundaries of substrings are then adjusted so that they abut good splice

donor and acceptor sites. Splice sites are also chosen in ways designed to avoid changes to the alignment.

Choose the best mappings. In the final step of the algorithm, each mapping will be assigned a score based on match reward, mismatch and gap penalties. The E-value is calculated based on [AG96] and [KA93]. The best mapping(s) will be selected based on the E-values for each DNA sequence.

The full algorithm used by ESTmapper is presented in Figure 2.1.

2.3.2 Mapping Statistics

As a metric for the accuracy of mapping results, most current DNA mapping algorithms [FHZ+98, Ken02, WCO01] simply report the percentage identity and coverage percentage for each mapping. As far as we know, current tools do not provide statistical significance estimates for mapping results similar to E-values produced by pairwise sequence alignment algorithms such as BLAST.

With only percentage identity and coverage percentages, it is difficult to distinguish two mappings with the same percentage identity and coverage percentages, but with different intron size (on genome) and gap size (on DNA sequence), both of which are very important for selecting the best mapping(s) since the best mappings tend to have the smallest intron size and gap size. To provide accurate mapping results, in ESTmapper we borrow and modify the "sum statistics" theory [AG96, KA93] from BLAST to assign an E-value for each mapping. We then use these to compare mappings and select the best mapping with the E-value as the

```
  // preprocess genome
  for each (subset) genome sequence
    build & store suffix tree T

  // map DNA sequence
  for each (subset) genome (if genome is partitioned)
  {
      load suffix tree T

      // find long common substrings and
      // build spliced/gapped matching regions
      for each DNA sequence E
      {
        // examine suffixes of E
        while offset F < length(E)
          for suffix of E beginning at offset F,
             find longest common substring S in T
          if length of S >= MIN_MATCH_SUBSTRING and
             frequency of S in genome <= MAX_MATCH_FREQ
             store S & its location in T
          increase offset F by MAX(length(S), MIN_SKIP)
        // extend long common substrings
        for each S
          extend it in both directions
        // build spliced/gapped mappings
        for all S
          examine locations of S1, S2
          if S1 near S2 with gap G <= MAX_GAP
            form/extend matching region M for S1,S2
          form mapping MAP based on M
        // refine mappings
        for each MAP
          refine mapping considering gaps and splice sit
          calculate alignment score and E-value
        assign to E the MAP with the smallest E-value
      }
  }

  find and output each DNA's best mapping location(s)
}
```

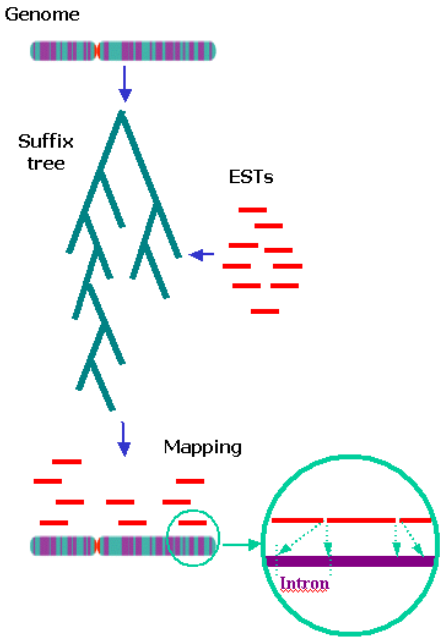

Figure 2.1: ESTmapper Algorithm

final output DNA-to-genome alignment.

According to [AG96], the optimal scores from ungapped local alignments follow an extreme value distribution. The probability that the optimal ungapped local alignment has a score S of at least x is:

$$P(S \geq x) = 1 - exp(-Kmne^{-\lambda x}) \tag{2.1}$$

where K is a constant given by a geometrically convergent series depending on score matrix and the probabilities of varies amino acid/nucleotide types, m is the number of letters in query sequence, n is the number of letters in database sequence(s) and λ is a positive scalar to convert the alignment raw score to a normalized score. The number of alignments expected by chance (E) during a sequence database search can then be estimated by:

$$E = Kmne^{-\lambda S} \tag{2.2}$$

These equations can also be extended to scoring gapped local alignments.

However, equations 2.1 and 2.2 only apply to the single highest score region of the sequence. When mapping DNA sequences to genome sequence, there are usually several high score regions because of the exon/intron structure in genome sequence. Under this condition, it is necessary to assess the combined statistics of multiple high score regions. If S_r is the rth highest score, the sum of the r highest score is

$$\begin{aligned} T_r = & \sum_{i=1}^{r}(\lambda S_i) - \ln(Kmn) - \\ & (r-1)(\ln(K) + \ln(G) + \ln(g)) - \log(r!) \end{aligned} \tag{2.3}$$

where G is the largest intron size in genome and g is the largest gap size in DNA sequence. ESTmapper calculated S_i for each extended long "common" substring

based on BLAST like local alignment score with match reward as 1, mismatch penalty as -3, gap opening penalty as -10 and gap extension penalty as -5. Then for large mn, the probability for T_r is:

$$f(t) = \frac{e^{-t}}{r!(r-2)!} \int_0^\infty y^{r-2} exp(-e^{(y-t)/r}) dy \tag{2.4}$$

To obtain the tail probability for $T_r \geq x$, ESTmapper will integrate $f(t)$ for t from x to infinity with the function from NCBI BLAST:

$$\begin{aligned} P_r(S \geq x) &= \frac{r^{r-2}}{(r-1)!(r-2)!} \int_x^\infty e^{-t} \\ &\quad \int_0^\infty y^{(r-2)} exp(-e^{(y-t/r)}) dy dt \end{aligned} \tag{2.5}$$

Usually, $P'_r(S \geq x) = P_r(S \geq x)/\beta^{(r-1)}(1-\beta)$ is used instead of $P_r(S \geq x)$ to correct the results for the statistic significance test. This is because only if $P_r(S \geq x)$ is less than $P(S \geq x)$ of any component high score region, these r region alignment is the optimal alignment and can be used to replace the P value of single highest score region. However, for mapping DNA sequence to genome sequence, since all the found exons should be part of the alignment, the correction function is not necessary. The final E value is calculated by $E = -\ln(1 - P_r)$. ESTmapper uses this E value to select the best mapping result.

Note that though we found it useful in ESTmapper to utilize both BLAST's statistical model for scoring matches and BLASTN's match extension algorithm for extending long common substrings, the underlying biological model for BLAST (homology, mutations) and ESTmapper (expressed DNA, splicing, sequencing errors) are different. We have been fortunate in that the two models are sufficiently similar

that ESTmapper appears to perform well in practice. It would be interesting to see whether adjusting both algorithms to DNA-to-genome mappings would improve mapping precision.

2.3.3 Implementation

ESTmapper is written in C and implemented on top of the WOTD suffix tree software (version 1.1) implemented by Stefan Kurtz [GJS03]. The WOTD suffix tree implementation was downloaded from `http://bibiserv.techfak.uni-bielefeld.de/download/tools/Wotd_11.html`. The WOTD suffix tree source code was modified and used as our main data structure for storing and indexing genomes.

2.3.4 Parallelization

Being able to exploit the power of parallel computing is a major advantage for computationally intensive applications such as mapping large number of DNA sequences onto genomes. It turns out that the ESTmapper algorithm, like many problems in computational biology, is embarrassingly parallel and can be easily parallelized at a coarse-grain level for efficient parallel execution. The parallel versions of ESTmapper work as follows.

2.3.4.1 Multithreaded ESTmapper

On shared-memory multiprocessors (SMPs) ESTmapper may be parallelized according to the master-worker parallel paradigm using pthreads. The master thread

first reads the suffix tree of the genome into memory sequentially. Next, the master thread spawns workers (pthreads) and assigns DNA sequences to each worker to find mappings. When all sequences have been mapped to the genome, the master thread collects and outputs the result.

Load balancing is simple since the bulk of the computation is mapping DNA sequences to the genome, and the mapping time is usually proportional to the number of bases being mapped. So the master thread just needs to assign roughly the same number of bases to each thread to ensure an even division of work.

2.3.4.2 Message-passing ESTmapper

On message-passing machines such as PC clusters, ESTmapper may be parallelized using the MPI communications library to communicate between processors. To reduce communication costs, each processor reads the suffix tree into memory independently. The master node assigns and sends DNA sequences each node. Every node then finds genome mappings for its sequences locally, without any need for interprocessor communication. Once mappings are computed for assigned sequences, each node sends its mappings to the master node to be collected and output.

ESTmapper is very efficient on clusters, since little communication is needed, just at the beginning and end of the overall computation to send out DNA sequences and retrieve their mappings. Performance can be improved further if each node stores a copy of the suffix tree in its local disk.

2.4 Experimental Results

2.4.1 Evaluation Environment

To evaluate ESTmapper, we compared its performance and precision with other mapping and clustering algorithms. We downloaded and installed the latest versions of BLAT, Sim4, Spidey, TGICL, PaCE, and the NCBI BLAST software suite and toolkit on a Sun SunFire 6800 SMP with 24 processors (UltraSparc III 750Mhz) and 72 GB memory. For message-passing speedups we also ran ESTmapper on a Linux PC cluster with 1.6 GHz AMD Athlon processors and 1 GB memory per node.

For test data, we downloaded EST, gene, and genome sequences for Arabidopsis Thaliana (mustard plant) and Homo Sapiens (human) from NCBI, UCSC genome browser and Sanger Institute. The Arabidopsis genome is about 115 million bases, and the Human genome is about 3 billion bases. The EST and gene DNA sequences we use were chosen because they were previously mapped to the genome by biologists (or put in widely accepted clusters). We can thus used them to evaluate the precision of mapping and clustering algorithms used by ESTmapper and other tools. Three data sets used in many of our experiments are:

- DataSet1: $263,255$ EST and cDNA sequences (average length 734 bases) from Arabidopsis UniGene build #44 from NCBI
- DataSet2: $28,952$ Arabidopsis genes (average length 1322 bases) and 5 Arabidopsis chromosomes from NCBI.

- DataSet3: 936 Human genes (average length 1936 bases) mapped to chromosome 22 and Build #30 of Human chromosome 22 from Sanger Institute.

- DataSet4: $1,383,818$ Human EST sequences (average length 686 bases) and Human genome Build #35 from NCBI.

2.4.2 Mapping Parameters

ESTmapper requires users to input basic mapping parameters. The default value for minimum common substring length is 25 bases, for maximum number of mapping locations is 30, for maximum human intron size is 500,000, for maximum Arabidopsis intron size is 6000, skip step size is 10.

2.4.3 Basic Performance

We begin by examining the time and memory required by ESTmapper with respect to suffix trees and multiple processors. WOTD suffix trees can be more expensive to build than other suffix trees in that they require $O(nlog(n))$ expected and $O(n^2)$ worst time to construct. However, they can yield good performance for substring searches because the WOTD data structures provide good cache locality. Since the WOTD suffix tree is stored in a flat array, users can also build the suffix tree once and store it on disk for later use.

When constructing a WOTD suffix tree for the genome, we need to select the number of trees we plan to use. ESTmapper can build a single suffix tree for the entire genome (by concatenating the DNA sequence of each chromosome), one

tree per chromosome, or any number of trees (by first concatenating, then splitting the DNA sequences of all chromosomes). Some additional bookkeeping is required to keep track of the original chromosomal locations of each sequence. When the genome is partitioned into multiple suffix trees, ESTmapper can iteratively compare nucleotide sequences against each suffix tree, record the mappings found, and select the best overall mapping at the end.

Figure 2.2a shows the result of using multiple suffix trees to map Arabidopsis ESTs from DataSet1 to the entire genome. We find ESTmapper runs faster with fewer suffix trees, but trees are larger and require more memory. We thus have a classic memory/speed tradeoff in choosing how to use ESTmapper. Choosing the proper number of trees to use depends on the available memory. If insufficient memory is available to hold the suffix tree for the entire genome, ESTmapper can reduce its memory usage at the expense of longer running times by partitioning the genome and building separate suffix trees for each portion of the genome.

Figure 2.2b shows the performance of multithreaded ESTmapper on the Sun-Fire 6800 SMP and message-passing ESTmapper on a Linux PC cluster when mapping DataSet1 to the entire Arabidopsis genome. We find that ESTmapper obtains fairly good 8-processor speedup for the SMP (6.9) and PC cluster (5.7).

We believe speedups are lower on the PC cluster because (sequential) file I/O to load suffix trees is both slower (44 seconds on PC cluster, versus 14 seconds on SMP), and takes up a larger portion of total execution time on the faster PCs (186 ESTs mapped/second on PC nodes, versus 83 ESTs mapped/second on the SMP). If the file I/O time is eliminated, ESTmapper achieves a 8-processor speedup of 7.5

on the SMP and 8.0 on the PC cluster. We thus expect ESTmapper speedups to improve for larger input data sets, where file I/O is less important. Memory use remains constant for ESTmapper relative to the number of processors, since each processor maintains a local copy of the suffix tree.

2.4.4 Genome Mapping Performance Comparison

The performance of six DNA-to-genome alignment tools (ESTmapper, BLAT, Sim4, Spidey, BLAST and megaBLAST) were evaluated with Arabidopsis and Human data. We mapped 936 Arabidopsis genes from DataSet2 onto chromosome I, and 936 Human genes in DataSet3 onto chromosome 22. All tools were run sequentially on a single processor of the SunFire 6800. Running times are shown in Figure 2.2c. "ESTmap" stands for ESTmapper and "megaB" stands for megaBLAST. Note that running times are presented using log scale along the Y-axis.

Results show that ESTmapper is the fastest among six evaluated software tools, and has reasonable memory usage. It is about 3–6 times as fast as BLAT, 21–45 times as fast as megaBLAST, 1000 times as fast as Spidey and 4000 times faster than Sim4. These results seem reasonable when we remember that only ESTmapper and BLAT preprocess the genome to improve performance. ESTmapper is likely more efficient than BLAT because it is able to find the longest common substrings directly, rather than extending hits.

Figure 2.2d presents the memory used by each of the alignment tools. We find megaBLAST and BLAST use the most memory, while BLAT requires the least

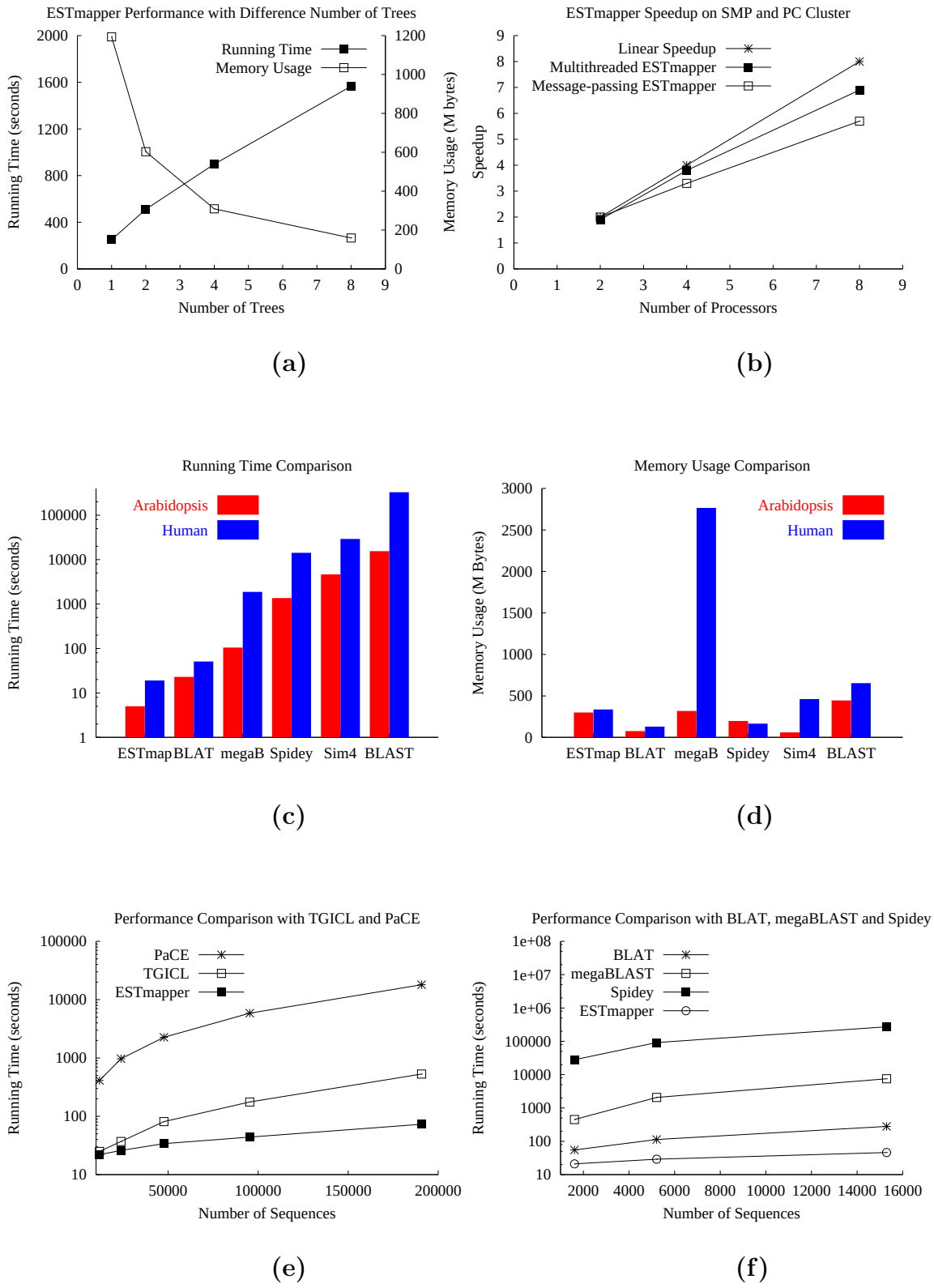

Figure 2.2: Performance Comparison of EST Mapping and Clustering Algorithms

memory. ESTmapper requires about 4 times more memory than BLAT.

2.4.5 Genome Mapping Precision Comparison

Next, we compared the precision of the DNA-to-genome mappings found by four tools (ESTmapper, BLAT, Sim4, Spidey) for all 28,952 Arabidopsis genes and 936 Human genes from DataSet2 and DataSet3. We used the Arabidopsis gene annotation information provided by biologists at TIGR and Human gene annotations from biologists at Sanger Institute as the correct mapping. Results about percentage of correctly mapped individual nucleotides, complete exons and genes are shown in Tables 2.1, 2.2 and 2.3.

Results show that Sim4 is the most precise mapping software, on both nucleotide level and exon level, and for both Arabidopsis sequences and Human sequences. On gene level, BLAT is most precise for Arabidopsis data (95.4% correct) and Sim4 is most precise for Human data (94.8% correct). So overall, for human sequences, Sim4 provides the best accuracy rate. ESTmapper is close behind, while BLAT and Spidey lag ($< 90\%$ correct on three levels).

	Spidey	BLAT	Sim4	ESTmapper
Arabidopsis	99.6%	97.3%	99.7%	99.4%
Human	80.5%	82.9%	99.9%	98.6%

Table 2.1: Precision Comparison with 36,298,530 Arabidopsis Nucleotides and 1,811,944 Human Nucleotides

	Spidey	BLAT	Sim4	ESTmapper
Arabidopsis	97.8%	97.0%	97.7%	96.1%
Human	86.6%	84.8%	98.0%	95.4%

Table 2.2: Precision Comparison with 155,970 Arabidopsis Exons and 5,176 Human Exons

	Spidey	BLAT	Sim4	ESTmapper
Arabidopsis	94.6%	95.4%	94.3%	90.6%
Human	87.4%	86.3%	94.8%	91.8%

Table 2.3: Precision Comparison with 28,952 Arabidopsis Genes and 936 Human Genes

Chromosome #	# of ESTs on chromosome	ESTmapper	BLAT
1	130,366	94.9%	90.3%
2	86,773	97.5%	92.4%
3	71,532	95.1%	91.8%
4	49,292	94.8%	95.1%
5	59.857	98.0%	92.9%
6	83,144	95.8%	89.9%
7	67,656	88.5%	41.8%
8	46,542	96.6%	93.5%
9	52,959	97.5%	92.7%
10	45,973	98.0%	93.1%
11	91,849	96.6%	93.7%
12	98,265	95.6%	96.5%
13	19,404	97.3%	93.0%
14	45,299	97.1%	93.0%
15	45,948	94.8%	91.9%
16	54,449	96.2%	91.6%
17	94,430	95.7%	87.1%
18	17,966	98.2%	91.7%
19	89,527	96.2%	92.0%
20	37,647	98.5%	95.7%
21	12,155	95.8%	92.7%
22	39,816	96.9%	92.3%
X	40,175	95.6%	92.8%
Y	2,794	97.7%	94.3%
Total	1,383,818	95.9%	89.1%

Table 2.4: Precision Comparison with 1.4 million Human ESTs by Chromosome

There is only one Human gene ESTmapper mapped incorrectly because there is one exon which is too small to be found with the default ESTmapper settings. Most such small exons can be found by ESTmapper with a finer grain search, at the cost of reduced processing speed. Fortunately, very small exons usually do not occur often in large genomes [DL99, SCK04]. For Arabidopsis sequences, Sim4 and BLAT are the most precise. ESTmapper provides the similar accuracy rate on nucleotide level, very close on exon level ($\sim$ 1% difference) and not far behind on gene level ($\sim$ 4% difference).

To further analyze precision of ESTmapper's mapping results, we compared ESTmapper and BLAT's mapping results on gene/EST level for all 1.4 million human EST sequences from DataSet4. The annotation information for these ESTs are obtained from NCBI Mapview database. We only used BLAT for comparison because of the processing speed limitation of other tools. Results are shown in Table 2.4. We find ESTmapper correctly mapped 96% of the 1.4 million ESTs to the human genome, while BLAT correctly mapped 89% of the ESTs, a difference of $94,350$ sequences. It seems ESTmapper has better precision than BLAT when mapping human ESTs.

While the precision of EST alignment is similar for most chromosomes, it is very different for chromosome 7. One possibility is that it has some very large introns, so it is hard for either tool to map all of the exons on the same EST sequence to the chromosome. Another possibility is that chromosome 7 has some unusual splicing motifs not found elsewhere, so it is difficult for either tool to detect exact exon boundaries. In either case ESTmapper still performs better than BLAT.

2.4.6 EST Clustering Performance Comparison

Next, we compared the performance and precision of using genome mappings to cluster ESTs against other clustering algorithms such as PaCE and TGICL. We form EST clusters out of all ESTs mapped to nearby or overlapping locations in the genome. For the comparison, we used genome mappings from ESTmapper, Spidey and BLAT.

In Figure 2.2e, we compared ESTmapper performance with TGICL and PaCE for 190,740 Arabidopsis ESTs mapped to the Arabidopsis genome preprocessed as a single suffix tree. All three tools were timed using 8 processors on the SunFire 6800. Note that running time is displayed in log scale along the Y-axis. We discover that ESTmapper is significantly faster for large numbers of ESTs. Figure 2.2f shows the performance of BLAT, Spidey, and megaBLAST when used to cluster ESTs using one processor on the SunFire 6800. Due to the slow processing speed of Spidey and high memory usage of megaBLAST, for the second set of experiment we were only able to use 15,293 EST sequences. We note once again that ESTmapper is the most efficient clustering algorithm.

We also found ESTmapper has almost constant memory usage of about 1 GB. In comparison, memory usage for PaCE and megaBLAST increases very quickly as the number of ESTs increases. PaCE used 1.4 GB memory on each processor when processing $190,740$ sequences. megaBLAST used about 4 GB memory when processing $15,293$ sequences and ran out of memory when the number of ESTs was increased.

2.4.7 EST Clustering Precision Comparison

To compare the precision of the clusters produced by the different algorithms, we used BLAT, Spidey, PaCE, TGICL and ESTmapper to cluster ESTs from Arabidopsis UniGene build #44 in DataSet1. Because of the memory limitation with PaCE and speed limitation with Spidey, we only used the 15,293 EST sequences from the first 1000 Arabidopsis clusters in UniGene. We measured the percentage of clusters exactly matching UniGene, the number of clusters produced by each algorithm, and the number of singleton clusters (with a single EST). The results are shown in Table 2.5.

	Spidey	BLAT	TGICL	PaCE	ESTm.
% identical	80.5%	81.4%	72.6%	60.5%	97.1%
# clusters	930	1152	1006	1575	1006
# singletons	–	101	296	573	9

Table 2.5: Precision Comparison with 1000 Arabidopsis UniGene Clusters

Again, ESTmapper produced clusters that are most similar (97%) to UniGene clusters. The other clustering techniques based on mapping ESTs to the genome were next (80%), while EST-only techniques produced significantly different clusters. ESTmapper also produced the smallest number of singletons clusters when compared to other algorithms. In addition to the results in Table 2.5, we also have clustering comparisons results for TGICL, BLAT and ESTmapper using the total 20640 Arabidopsis UniGene clusters. Among the three algorithms, TGICL found

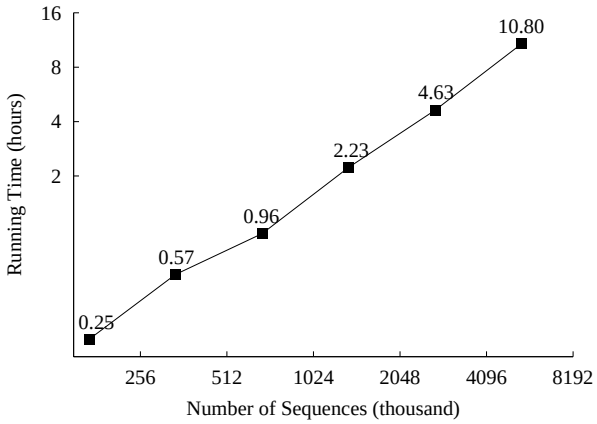

Figure 2.3: Clustering 5.5×10^6 Human ESTs

53.8% exactly matched clusters, BLAT found 66.5% exactly matched clusters, and ESTmapper found 83.6% exactly matched clusters. Though all three algorithms are less precise for the full set of Arabidopsis UniGene clusters, ESTmapper still provides the results closest to UniGene.

2.4.8 Scalability

Finally, to evaluate the scalability of ESTmapper, we also measured the computation time and memory usage of ESTmapper when used (on a SunFire 6800 with 8 processors) to cluster all 5.5 million human EST sequences by mapping them against the human genome (Build 35). To reduce memory use, the genome was split into 30 equal-sized pieces requiring about 1 GB suffix trees for each piece. Results are shown in Figure 2.3. ESTmapper was able to cluster all ESTs in 10.8 hours (0.056 seconds per EST for each processor), with the processing time increasing fairly linearly with the number of ESTs. Performance can be improved by using more memory and fewer trees.

2.4.9 Discussion

Our experimental evaluation of ESTmapper shows it is quite efficient and precise when compared with other sequence alignment techniques such as BLAT. For the purpose of EST clustering, ESTmapper also performed very well compared to other clustering algorithms.

2.5 Conclusion

Our experimental evaluation of ESTmapper shows it is quite efficient and precise when compared with other sequence alignment techniques such as BLAT. The only disadvantage is that ESTmapper requires more memory and disk storage to hold and store the suffix tree needed for the genome. Fortunately the amount of memory and disk space available in computers is quickly increasing while genome size stays constant, so memory use should become less of an issue as time goes on. For the purpose of EST clustering, ESTmapper also performed very well compared to other clustering algorithms. The main disadvantage here is that ESTmapper can only be used to cluster ESTs for organisms with sequenced genomes. As a result ESTmapper will probably serve simply to complement tools such as TGICL and PaCE.

Chapter 3

Mass Spectrometry Based Peptide Identification

3.1 Background

Mass spectrometry (MS) is an analytical technique used to identify chemical composition of the sample. More recently, with the advance of high throughput equipment design, it has been used to identify protein content of biological samples. Tandem mass spectrometry technique (MS/MS), when used in conjunction with liquid chromatography (LC), can quickly determine the protein content of biological samples in wide variety of context.

Since automated high-throughput mass spectrometry equipment can generate huge amount of mass spectra data on a daily basis, analyzing results using efficient and precise computer tools becomes an important step in mass spectrometry based peptide identification. Despite presence of large amounts of noise in mass spectra data, researchers have successfully used many different statistical models and computer algorithms to improve peptide identification accuracy.

3.1.1 Mass Spectrometry

Mass spectrometers work by splitting biological samples into collections of charged gas particles (ions) with different mass/charge ratios, then measuring their relative abundance by magnetically accelerating the ions and sending them to a

detector. Information on the mass of all ions detected can then be used to identify the chemical composition of the original sample.

In tandem mass spectrometry, ions are fragmented using techniques such as collision-induced-disassociation (CID), which shoots ions at high speed through a cloud of inert gas particles. The ions resulting from collisions are fragments of the original (precursor) peptide fragment, and can be used to identify its amino acid sequence.

The peptide fragmentation pattern usually depends on the physical and chemical properties of the peptide sequence. Because the CO-NH bonds connecting amino acids are weak, they tend to break most frequently when a peptide is fragmented. Ions containing the N-terminus are known as b_i ions, where i indicates the number of amino acids composing the ion. Similarly, fragments containing the C-terminus are labeled as y_i ions.

Almost all peptide identification algorithms rely heavily on identifying b and y ions in mass spectra. Other types of ions may also be created during fragmentation (a and x ions if the CH-CO bond is broken, c and z ions if the NH-CH bond is broken), but with much lower probability since those bonds are stronger than the CO-NH bond.

The mass spectrometer measures the mass-to-charge ratio (m/z) of ionized gas phase molecules. When the input are protein molecules, peptides are ionized and their m/z values measured. The measured m/z values of the peptide fragment ions form the so called tandem mass spectrum of the peptide, as shown in Figure 3.1. In the figure, the x-axis is the mass/charge ratio of the ions and the y-axis is the

intensity of the ions (number of particles detected).

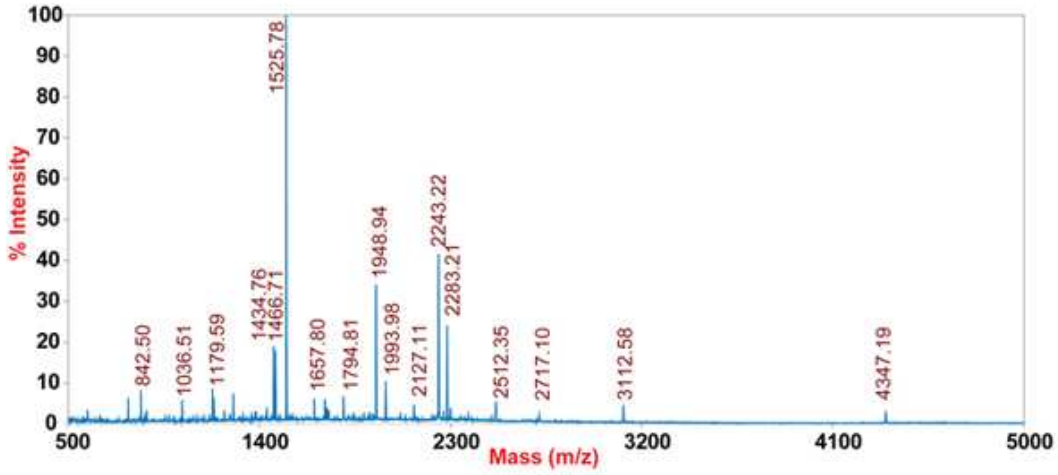

Figure 3.1: Mass Spectrum

In a high-throughput setting, a complex mixture of unknown proteins are cut into short amino acid sequences (peptides) using a digestion enzyme such as trypsin; fractionated into reduced complexity samples on the basis of some physical or chemical property such as hydrophobicity; and then a tandem mass spectrum is taken for selected observed peptides in each fraction. The end result of such an experiment is a set of a few hundred to a few thousand tandem mass spectra, each of which represents a peptide of about 6-20 amino acid residues. Typically, amino acid sequences of 8-10 residues carry sufficient information to determine the protein from which the peptide is derived. This experimental protocol can reliably identify hundreds of proteins from a complex mixture in several hours of instrument time.

3.1.2 Computer Algorithms for Peptide Identification

With recent developments in peptide identification algorithms, the mass spectra interpretation process has been automated. Now the effectiveness of mass spec-

trometry is determined by the ability of software tools to correctly identify peptides. There are three main classes of software algorithms for mass spectra based peptide identification. The first class of algorithms are designed to identify peptides and proteins through protein database searches. The second class of algorithms are designed to identify peptides and proteins using de novo sequencing techniques. The third class of algorithms are designed to detect peptide through spectral matching.

3.1.2.1 Database Search Based Peptide Identification

The most popular approach for identifying peptides using mass spectra is based on searching protein databases. Sequence database search algorithms [YECB96, PPCC99, CB04, GMK$^+$04, TSF$^+$05] use protein sequence databases to suggest peptide candidates for each spectrum. The peptide identification algorithms in category generally follow three steps to identify peptides.

The first step is to computationally digest protein sequences with specific enzymes. In another word, the peptide identification algorithms predict what peptides can be produced if the protein is entirely or partially digested by an enzyme. Then the mass of the peptides are computed and compared with the spectrum mass. Only those peptides whose mass falls within specified mass tolerance are selected as candidate peptides for the next algorithmic step. In the second step, the algorithms create a hypothetical spectrum for each peptide candidate by predicting the peptide fragmentation pattern and computing the masses of the fragments. The third step is to compare the experimental spectrum with the list of hypothetical spectra.

The algorithms use different match metrics (such as number of matched peaks, number of matched ions, mass difference, etc.) to measure the similarity between experimental spectrum and hypothetical spectrum. A score is generated to summerize the comparison results. For each experimental spectrum, a list of potential peptides are ranked according their scores. The highest score peptide is usually considered as the one that matches the experimental spectrum most closely. The database search based peptide identification approach has been very successful in practice.

SEQUEST [YECB96] is one of the first algorithms to identify proteins by correlating peptide tandem mass spectrum with amino acid sequence in protein database. It constructs the hypothetical spectra by predicting the mass-charge ratio of the fragment ions. The relative intensity of the ions is assigned according to authors' empirical knowledge of the appearance of tandem mass spectra for peptides. For mass-charge ratios that represent b ion or y ion, it assigns a magnitude of 50.0; for mass-charge ratios that are within 1 distance to b or y ions, it assigns a magnitude of 25.0; for mass-charge ratio that represent neutral losses of ammonia, water, carbon monoxide and those within 1 distance, it assigns a magnitude of 10.0. It then compares the experimental mass spectrum with hypothetical spectrum based on a cross-correlation function between the two spectra.

Mascot [PPCC99], developed by Perkins et al., is one of the most widely used protein identification softwares. It is based on MOWSE [PHB93] algorithm with addition of probability based scoring. MOWSE algorithm considers the relative frequency of a peptide with a given molecular weight being within a protein that

falls in a given range of molecular weights. Mascot algorithm is based on MOWSE, it uses additional probability-based score. The details of the algorithm are not released to the public. But the authors claim the absolute score is the probability that a match between experimental spectrum and peptide is a random event. To avoid the confusion when directly reporting absolute probability score, Mascot reports the scores as $-10\log_{10}(P)$, where P is the absolute probability score.

OMSSA [GMK+04] is a probability-based protein identification method from NCBI. After selecting the peptide candidates, it calculates a score based on the statistic significance of a match. The basic assumption used by OMSSA is that the number of product ion matches follows a Poisson distribution. It calculates an E-value based on the number of random matches against N theoretical spectra.

X!Tandem [CB04] is a multi-step algorithm for quick peptide identification from mass spectra. It filters out sequence candidates in multiple searching steps, while considering more stringent searching criteria in each step. The central assumption used by X!Tandem to improve the performance of filtering is that for each identifiable protein, there is at least one detectable tryptic peptide. It can thus use a quick match algorithm to look for fully digested tryptic peptides, and then look for a more comprehensive list of peptides using heuristics limited to the proteins where at least one match has been found.

The score function used by X!Tandem is based on a hypergeometric distribution calculated as the dot product of the intensities of the matching ions, multiplied by the factorials of the number of matched b and y ions. The E-value calculated by X!Tandem is based on just how unlikely a greater hyperscore is to be found, based

on statistical analysis of current hyperscores.

InsPecT [TSF+05] is an algorithm that performs high-throughput identification of peptide mass spectra. Its emphasis is on efficiently identifying post-translational modifications and mutations with high confidence. It uses peptide sequence tags as filters to select a small number of database sequences that contain the peptide which produced experimental spectrum with very high probability. InsPecT is able to search a broad range of post-translational modifications efficiently by searching within the largely reduced search space.

PepHMM [WYC06] is an algorithm that uses HMMs for peptide identification. It builds a HMM to capture the correlation among matched and unmatched ions to improve the precision of peptide identification.The proposed HMM can be used as a general post-process model for any experimental spectrum to theoretical spectrum matching scheme. Once trained, HMM can be used to reassess spectra comparison results.

One weakness of database search engines is that they do not fully use peak intensity information in the mass spectra when computing scores. An even greater flaw of database search based peptide identification is that the results are limited to known protein sequences in the database. It is thus impossible to identify new protein sequences using database search engines. Researchers are trying to compensate this by incorporating six-frame translation of nucleotide sequences (EST, SNP, etc.) into the protein sequence search space. An alternative relies on de novo sequencing.

3.1.2.2 De Novo Sequencing

De novo peptide identification algorithms [DAC+99, BE03, CKT+01, MZL+03, TJ97] attempt to determine the peptide sequence using only the peptide fragment information of the tandem mass spectrum. Candidate peptides are generated without using protein sequence databases. The typical de novo algorithms starts by creating a spectrum graph for each experimental spectrum after filtering the noise peaks. The spectrum graph is a graph representation of experimental spectrum.

The application of graph theory in de novo sequencing was first proposed by Bartels [Bar90]. In the spectrum graph, each peak in the experimental spectrum is represented as a vertex (or several vertices). The algorithms connect two vertices with an edge if the mass difference between two vertices equals the mass of an amino acid (within some mass tolerance). The edge is then labeled with the name of the amino acid. The weight of the edge is given by a score function that measures the mass difference between the theoretical mass and the experimental mass of the edge.

Using the de novo approach, the problem of peptide identification then becomes the problem of finding the longest path in the spectrum graph. Some algorithms also use spectrum graphs to generate possible peptide sequences. The peptide sequences set is extended and trimmed in an iterative procedure based on the design of the algorithm. A different class of algorithms use score functions to optimize over the space of possible peptide sequences.

Researchers in the field use graph theoretic algorithms, Markov chain Monte Carlo heuristic optimization, or HMMs to perform de novo peptide identification.

The advantage of de novo algorithms is that it works without requiring known protein sequences. De novo algorithms can thus be used to discover novel peptides.

Tools such as GutenTag [TSY03] demonstrate other advantages of de novo algorithms. First, by accounting for mass differences between vertices due to post-translational modifications (PTMs), de novo algorithms can detect many PTMs in a computationally efficient way. Second, de novo algorithms may be able to detect very short peptide sequences (tags) with high confidence, even if the entire peptide sequence cannot be determined, providing useful information to complement other peptide identification methods.

A weakness of de novo sequencing is that the percentage of spectra in a data set that can be identified conclusively using novo analysis is usually quite low, because tandem mass spectra often do not contain enough fragments to draw unambiguous conclusions. As a result, de novo peptide identification techniques only work well with high quality mass spectra. In addition, since the observed peptide sequences represent only a small portion of possible amino acid sequences, de novo interpretations typically produce a long list of equally possible peptide sequences.

Some researchers have proposed methods that combine both de novo sequencing techniques and a database search based peptide identification strategy. Mann et al. use a sequence tag of three or so amino acids derived directly from the tandem mass spectrum to search a protein sequence database [MW94]. This approach has not been widely adopted due to the lack of an automated technique for sequence tag derivation.

3.1.2.3 Spectral Matching

The spectral matching approach identifies peptides by comparing new experimental mass spectra with a reference mass spectra library. The mass spectra library is built with previously observed and identified spectra. Spectra in the library usually represent a consensus of several experimental mass spectra that have been identified and mapped to the same known peptide. Spectral matching is different from both database search based and de novo sequencing methods in that it makes no attempt at predicting the unknown spectra. Instead, it compares the unknown spectra with library spectra – the observed spectra consensus.

One advantage of spectral matching approach is it can use more information in each mass spectrum, such as the peak intensity information observed in previously identified spectra. In comparison, both database search based and de novo sequencing peptide identification algorithms usually consider peak intensity to be a minor factor when matching experimental spectrum with hypothetical spectrum generated from protein sequence. One reason is the unpredictability of the peak intensity. The unknown ion types, missing ions, noise, isotopic ions and machine errors all make peak intensity vary from run to run. However, with identified spectra, it is much easier to use peak intensity, which can contains important information about the peptide sequence.

With both ion mass and peak intensity information, spectral matching may be able to make peptide identifications with higher confidence than by using mass alone. Under some circumstances, a spectrum with only a few peaks can be strongly

identified because of the fragmentation characteristic of that peptide. In contrast, a small number of peaks can only provide very limited information for traditional database search based methods to disambiguate similar peptide sequences.

Traditional spectral matching approaches use a library of spectra and a spectral similarity measure to evaluate the quality of the spectrum-spectrum match [Ste94, SS94, Ste95]. Library spectra can be real spectra, identified using traditional sequence database search tools, or consensus spectra that summarize the most important features of spectra from a particular peptide. Spectral comparison functions have traditionally been based on the dot-product of a vectorized representation of each spectrum.

Recently, a number of groups have described spectral libraries of high-quality real spectra with high-confidence identifications [YMG+98, JCL+06, FMW+06, LDE+07] and consensus spectra [SKN+06, LDE+06, CCFB06]. Results indicate spectral matching can be both precise and efficient, identifying many spectra missed database search based algorithms. NIST's MS Search software [MSS05] is a dot-product based spectral matching search engine, it searches libraries of peptide fragmentation spectra [SKN+06] to identify unknown mass spectra.

However, the traditional dot-product and consensus based spectra matching approach do not fully utilize all the information contained in the spectra. The dot product, for example, weights the contribution of each spectra peak only by its intensity, without considering its consistency. Using a consensus spectrum assumes that a single intensity value is sufficient to represent the range of intensities observed in the experimental spectra. So a probabilistic model that can capture the spec-

tra peak intensity consensus and variation pattern can potentially provide better accuracy.

3.2 HMMatch: Peptide Identification by Spectral Matching of Tandem Mass Spectra Using Hidden Markov Models

As described in section 3.1.2, the traditional peptide identification algorithms tend to ignore the ion peak intensity information or only use partial intensity information. Even though some algorithms [BE01, SKSS03, Zha04, WYC06] have tried to predict the probability of observing an ion or its intensity, when applied to analyzing real life experimental spectra, the improvement made by these algorithms seems limited.

Based on experimental observation, a hidden Markov model (HMM) [Rab89] approach for spectral matching is designed and implemented. The approach uses hidden Markov models to summarize the peak intensity consensus and variation of example peptide fragmentation spectra. The unknown mass spectrum can be identified by comparing with these peptide mass spectrum HMMs.

3.2.1 Related Work

3.2.1.1 HMM and Its Application to Peptide Identification

The HMM [Rab89, JR90, JR85] was initially introduced in late 1960s and early 1970s and was used for human speech recognition. It includes a class of statistical

models used to capture statistical characteristics of time series and linear sequences. HMM is basically a double embedded stochastic process with first order Markov chain as underlying (hidden) stochastic process and state dependent probabilistic function as the observable stochastic process. Because of the nice mathematical properties of HMM, it was introduced into computational biology in late 1980s and used to profile protein sequences [Edd98, KBM$^+$94]. Later, HMM was also used to model gene structure and predict protein structure by fold recognition.

Researchers have used HMM for peptide identification only recently. PepHMM [WYC06] is one of these algorithms. It builds a HMM to capture the correlation among matched and unmatched ions, which has not been proposed by any other researcher. The proposed HMM is a general post-process model for any experimental spectrum to theoretical spectrum matching scheme, and it is used to re-access experimental spectrum vs. theoretical spectrum comparison results. However, the proposed model only distinguishes ions at the start, end and middle positions. As shown by the preliminary results in the following results section, the intensity patterns for ions change from peptide to peptide, so using one model can over-generalize the hidden intensity information.

Other researchers have proposed a generative HMM of mass spectra for de novo peptide sequencing called NovoHMM [FRR$^+$05]. The model emulates the whole mass spectra generation process in a probabilistic way. The results show NovoHMM outperforms other de novo sequencing algorithms that use mass spectrum graphs for generating candidate peptides. Researchers in signal processing area have also used HMMs or similar models to correct mass spectra alignment along time axis, so

as to extract the non-noise peaks using multiple alignment of continuous time series [LNRE05, LE05].

3.2.2 Motivation

The motivation for using HMM for spectral matching starts with multiple sequence alignment with HMM. Because of the similarities existing between two problems, we found it natural to adapt the probability model and apply it to spectral matching for tandem mass spectra.

3.2.2.1 HMMER for Multiple Sequence Alignment

Hidden Markov models have proved to be a powerful technique in statistical machine learning, and have been used extensively in *de novo* gene finding and protein family clustering and prediction [KBM$^+$94, Edd98, FMSB$^+$06]. In particular, the use of hidden Markov models for protein families, as in Pfam [FMSB$^+$06], suggests that they may find application in spectral matching. Given a multiple alignment of protein sequences that belong to the same protein family, the Pfam hidden Markov model represents a statistical consensus for the amino acids observed at each position. Conserved positions tightly constrain protein family membership, while the contribution of highly variable positions are down-weighted. The model also permits the insertion or deletion of amino acids to align diverged sequences at positions of significant conservation. The Pfam hidden Markov model, as a statistical signature of a protein family, is more sensitive than sequence alignment with any single mem-

ber of the family, and more specific than aligning against all protein family members. Because of the similarity between statistical consensus of amino acids observed at each position and statistical consensus of peak intensity at differetn m/z position, it motivates us to use HMM to model mass spectra peak intensity consensus and variation.

3.2.2.2 MS/MS Intensity Profile

Figure 3.2 summarizes the statistical properties of the low-mass region of a set of spectra. These spectra are identified as peptide DLATVYVDVLK with high confidence. The figure shows the distribution of normalized peak intensities before $\log_{10}$ intensity transformation with box-plot. Regions denoted $I_0, I_1, \ldots$ represent m/z value regions between singly charged b and y ions of peptide DLATVYVDVLK, while the regions denoted $b_1, b_2, \ldots$ and $y_1, y_2, \ldots$ represent m/z values within 0.5 Da of the corresponding b and y ions. Above each box-plot is the average frequency of selected peaks, per spectrum, in each m/z region. While there is a lot of consistency in the intensity of some ions, other ions' intensity show considerable variation, and some b and y ions are completely absent. The peak intensity consensus and variation here is very similar to the consensus and variation of amino acids at each multiple sequence alignment position. So the hidden Markov models are used to capture this complexity peak intensity pattern for each peptide.

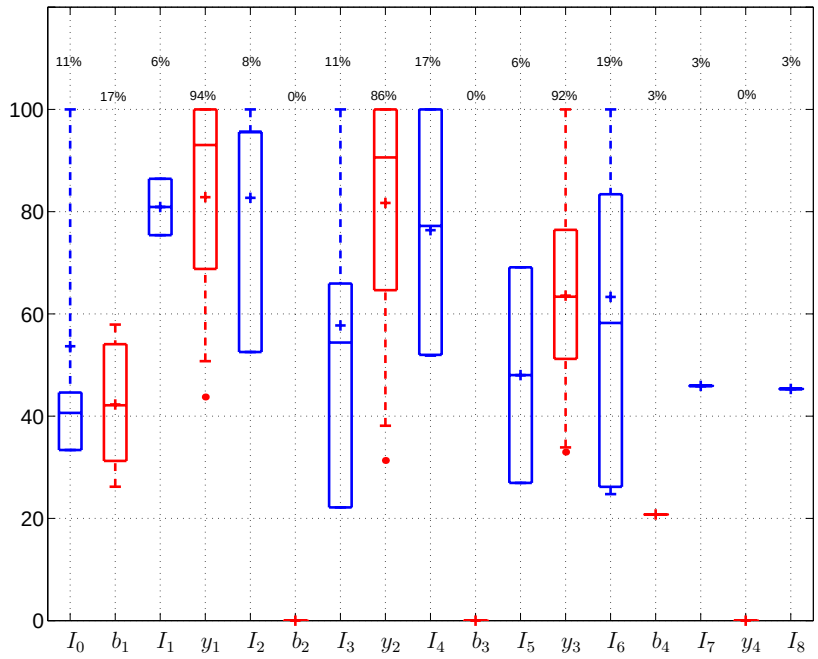

Figure 3.2: Normalized intensity distribution (before $\log_{10}$ transformation) of low-mass peaks in m/z regions between $(I_0, \ldots)$ and near singly charged b $(b_1, \ldots)$ and y $(y_1, \ldots)$ ions from training spectra for peptide DLATVYVDVLK. Average peak frequency, per spectrum, in each m/z region is indicated above each normalized intensity box-plot.

3.2.2.3 HMM for Spectral Matching

The hidden Markov models are trained with a set of mass spectra that are confidently assigned to a specific peptide X!Tandem [CB04]. Then the hidden Markov models are used to evaluate additional mass spectra, and the results compared with X!Tandem, Mascot and NIST's MS Search software. This approach is different from PepHMM [WYC06], which uses a single hidden Markov model to evaluate the quality of any peptide-spectrum match. The results show that the HMM MS/MS Match (HMMatch) approach identifies many mass spectra left unidentified by Mascot and X!Tandem. It is also more flexible and robust than MS Search. In the following sections, the HMMatch design and the experimental evaluation of HMMatch for peptide identification are described in details.

3.2.3 Training Data Selection

Both training and testing data (mass spectra) are processed and provided by Professor Nathan Edwards. The following paragraphs are data selection method described by Professor Nathan Edwards.

3.2.3.1 Mass Spectra Libraries

Public LC/MS/MS data sets derived from human samples are downloaded from their respective data repositories. Sources include PeptideAtlas [DDK+06] and Human Proteome Project (HUPO) Plasma Proteome Project (PPP) [OSA+05]. In all, there are more than 2.3 million spectra stored locally for searching. These 722

data files represent work from 33 different labs or projects. Tandem mass spectra are re-searched using conservative search parameters: 1 missed cleavage, tryptic N and C-terminals, methionine oxidation and cysteine alkylation modifications only, precursor mass tolerance 2 Da, and fragment mass tolerance 0.4 Da. The computation is done with a computational grid of approximately 250 Linux CPUs managed by the Condor scheduling infrastructure, and the X!Tandem search engine. The Mascot search engine [PPCC99], tied to a single processor, is used for benchmarking, comparison studies, specific identifications confirmation.

3.2.3.2 Data Selection

The results of re-searching public LC/MS/MS spectra are stored in a relational database. A small set of frequently observed peptides (in a particular charge state) with many high-confidence peptide identifications are chosen. One HMMatch model is constructed for each of these peptides. For each peptide model, the set of MS/MS spectra extracted are with precursors within 2 Da of the peptide's theoretical monoisotopic m/z value.

These spectral sets are partitioned into five classes based on their X!Tandem search results: High confidence identifications of the model peptide (denoted HC), low confidence and non-significant identifications of the model peptide (LC), high confidence identifications of a non-model peptide (HC-Other), low confidence and non-significant identifications of a non-model peptide (LC-Other), and spectra with no peptide identification with E-value less than 1 (Unknown). A 10^{-4} X!Tandem

E-value threshold is used to delineate high-confidence identifications from other peptide identifications. Approximately half of the model peptides' high confidence identifications are randomly selected for training (HC-Train), and the remaining peptides are reserved for testing (HC-Test).

However, there is a problem when extracting evalues from search results. While the E-values of high-scoring peptide identifications of model peptides are easy to extract from Mascot and X!Tandem search results, E-values from low-scoring and non-significant scores are more difficult to obtain, because the search engines output only the top few best scoring peptides for each spectrum.

To obtain E-values with respect to the model peptides of weak identifications, a special decoy protein sequence database is constructed following [EHFG05a]. This sequence database consists of reversed protein sequences of the IPI-Human [KDW$^+$04] protein sequence database, plus (forward) protein sequences that contain the model peptides. The size of this decoy database is consistent with IPI-Human, and high-confidence identifications by X!Tandem and Mascot have similar E-values as for a IPI-Human search.

The X!Tandem source code is modified to output all peptide identifications with hyperscores within 80% of the best scoring peptide. Lastly, the precursor mass tolerance parameter is increased to 4.5 (6.5) Da for spectra from charge 2 (3) peptide spectra data sets. Using these techniques, it is able to obtain valid E-value estimates for many spectra with respect to the model peptides, even when the identifications are very weak or missing from our original re-search results. These E-value estimates are later used in Section 3.2.8.4.

3.2.4 Spectra Pre-processing

3.2.4.1 Spectra Normalization

While the fragmentation spectra of peptides that subject to collision induced dissociation are widely believed to be reproducible under a variety of experimental and instrumental conditions, this reproducibility is difficult to observe without considerable normalization of each spectrum. As described in [Ste94, SS94, YMG+98, WYC06, CCFB06, FMW+06, LDE+06, SKN+06, LDE+07], a number of techniques have been proposed to pre-process and normalize the spectra, whose peaks are represented by a list of positive real-valued (m/z,int) pairs, ordered by m/z. These techniques include intensity normalization relative to the base peak or rank, m/z binning and blurring, transformations such as the square root or logarithm of peak intensity, and elimination of insignificant ions by intensity or ranking. However, there seems to be little consensus, to date, on spectrum normalization techniques for spectral matching, although the work of [WLM+05] provides some basis for evaluating many possibilities.

For the proposed spectral matching framework, a number of normalization procedures were tested and the procedure which produced good results was selected to pre-process spectra data, although the selection is not based on comprehensive examination of all the possibilities. This issue will be revisited in future work. The following spectrum normalization techniques used by HMMatch are restricted to those can be carried out without knowledge of the model peptide or global properties of the training spectra, with the hope that these techniques might be useful in other

settings.

Peak Intensity Normalization. The absolute value of peak intensities varies significantly between mass spectra of the same peptide, due to peptide abundance variation and different instrument technologies. The intensity of each peak in the peak list is normalized by the 3rd most intense peak. The normalized intensity of the 1st and 2nd peaks are reset to 100%. The 3rd most intense peak, rather than the base peak, is used to avoid creating spurious variation due to a non-reproducible base peak (often the precursor ion).

Peak Selection. To ensure that only the most informative peaks are used in HMMatch training, only the top 10 peaks are kept with normalized intensity at least 1%. The selection of a small number of peaks ensures that the significant variation in the number of small, insignificant, "grass" peaks in each spectrum is eliminated. The resulting fixed length peak list ensures that HMMatch scores for different spectra are consistent in magnitude. This aggressive peak selection does not compromise HMMatch's performance.

Intensity Transformation. Normalized intensities are transformed using the base 10 logarithm, which helps to moderate the larger variance observed in more intense peaks. After transformation, retained normalized intensity values range from 0 to 2.

3.2.4.2 Spectra Discretization

Normalized peak intensities are discretized with respect to 4 equal-sized bins between the minimum and maximum normalized intensity values. For the spectrum normalization described above, normalized intensities range from $[0, 2]$, resulting in normalized intensity bins $[0.0, 0.5]$, $(0.5, 1.0]$, $(1.0, 1.5]$, and $(1.5, 2.0]$. We denote these intensity bins $\mathcal{I}$.

Each m/z value in the peak list is transformed from a positive real-valued number to a symbol describing a region of the m/z axis. Let $M = (m_1, \ldots, m_k)$ be an ordered sequence of theoretical m/z values of expected or commonly observed ions, and ε be a suitable mass tolerance for matching observed peaks with the elements of M. The values $m_0 = -\varepsilon$ and $m_{k+1} = +\infty$ are added to M for notational convenience. The m/z axis is then partitioned into $3k+1$ regions: $R_j^L = [m_j - \varepsilon, m_j]$ for $j = 1, \ldots, k$, $R_j^H = (m_j, m_j + \varepsilon]$ for $j = 1, \ldots, k$, and $R_{j,j+1} = (m_j + \varepsilon, m_{j+1} - \varepsilon)$ for $j = 0, \ldots, k$. Figure 3.3 shows these regions on the m/z axis. $\varepsilon = 0.5$ Da is used throughout. The set of all such regions are denoted as $\mathcal{R}$.

3.2.5 HMMatch Algorithm

The HMMatch hidden Markov model for peptide fragmentation spectra is based on the protein family hidden Markov model used by the Pfam database, and is shown in Figure 3.4.

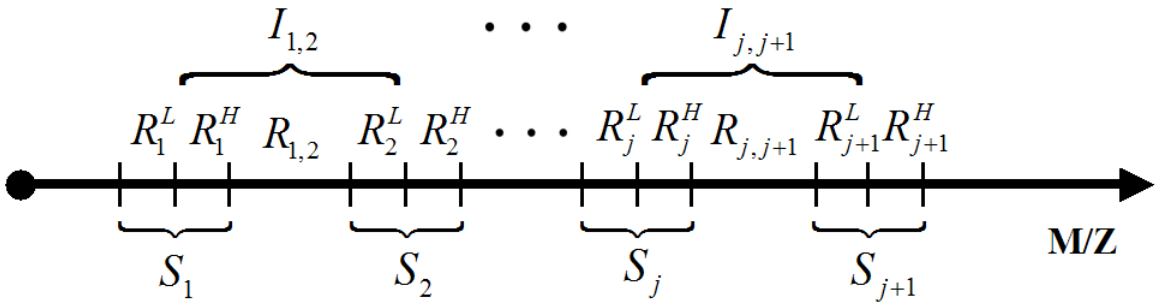

Figure 3.3: Discretization of m/z axis into regions. Valid m/z region emissions, for each non-silent HMM hidden state, also shown.

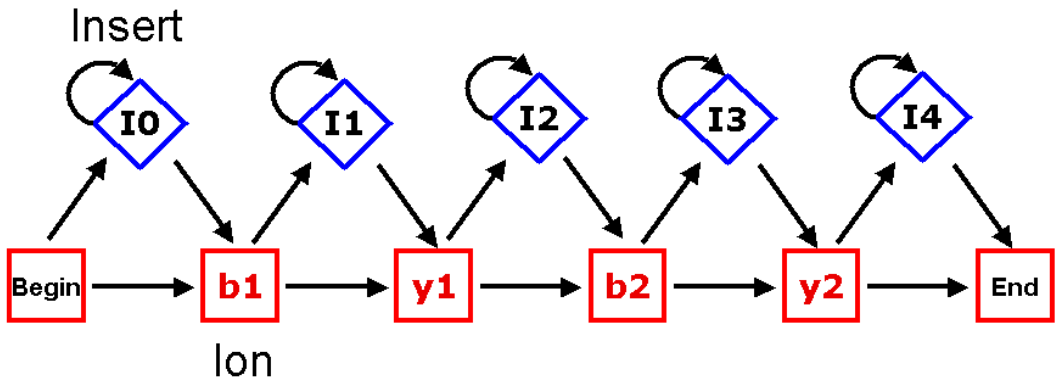

Figure 3.4: The HMMatch hidden Markov model for peptide fragmentation mass spectra.

3.2.5.1 Hidden States

The hidden Markov model represents each of a peptide's commonly observed or expected ions, with m/z value $m_j \in M$, by a hidden state S_j. Initially states only represent singly charged b and y ions for 2+ peptides. But later it is found that by adding some doubly charged b and y ions and dropping rarely observed ions, the predictive performance for 3+ peptides can be improved. These hidden states, represented by squares in Figure 3.4, are ordered as in M. Insertion states $I_{j,j+1}$ (diamonds in Figure 3.4) represent additional, unexpected, or unmodeled ions between expected m/z values $m_j, m_{j+1} \in M$. The absence of expected m/z values is modeled by zero emission of intensity for ion states.

3.2.5.2 Emission Probabilities

Each non-silent state emits a discrete valued m/z value-intensity pair $(m/z, int)$ from $\mathcal{R} \times \mathcal{I}$. The m/z value and intensity emission probabilities are modeled as independent, so that $P_S(m/z, int) = P_S(m/z)P_S(int)$ for each non-silent state S. Each ion state S_j, for $m_j \in M$, emits peaks such that $m/z \in \{R_j^L, R_j^H\}$. Similarly, each insertion state $I_{j,j+1}$ between states S_j and S_{j+1}, for $m_j, m_{j+1} \in M$ emits peaks such that $m/z \in \{R_j^H, R_{j,j+1}, R_{j+1}^L\}$. Figure 3.3 shows the valid m/z regions that may be output by each non-silent state.

3.2.5.3 Transition Probabilities

Non-zero transition probabilities are shown as directed edges in Figure 3.4.

3.2.6 Statistical Significance of HMMatch Score

3.2.6.1 HMMatch Score

Once trained, the HMMatch hidden Markov model is used to assess the extent that unknown spectra are consistent with the training spectra. Unknown spectra are normalized and discretized, as above, and fed into the trained models. There are several metrics that can be used to measure probability that the model generate the given sequence: the Viterbi distance and Forward distance. The Viterbi algorithm is used to compute the probability of the most likely path through the hidden Markov model. The Forward algorithm is used to probability of a particular output sequence. For HMMatch, both metrics are tested as HMMatch score, which equals to $-\log_{10}(p)$, where p is the probability of the Viterbi path or the probability of Forward probability. The experimental results will show that both metrics have similar behavior for the selected data sets.

However, because the HMMs are different for different peptide mass spectra patterns, HMMatch score cannot be used for the comparison across different models. Random spectra are needed to normalize HMMatch score and give statistical significance for the spectra identification results.

3.2.6.2 Random Spectra

Generating random spectra for statistical significance models must be done with care. Naively generated random spectra are so unlike peptide MS/MS spectra that even poor HMMatch scores appear statistically significant. To provide good

statistical significance estimation of the results, it is required that the random spectra look enough like true peptide fragmentation spectra so that poor HMMatch scores are not statistically significant, while good HMMatch scores are still unlikely to be observed.

The intensity and m/z value properties of discretized, normalized training spectra are extracted independently. Each peak list contains 10 peaks, 3 of which have normalized intensity 100%, by construction. The discrete intensities of the remaining peaks in each training spectrum are tabulated to construct an empirical probability density for peak intensity.

The probability density of the discretized training spectra m/z values is determined from the m/z values of the top N most intense peaks in each spectrum, for some $N \geq 10$. As with the intensities, the m/z regions are tabulated to construct an empirical probability density. For small N, the probability density strongly favors m/z regions corresponding to very abundant ions, while for large N, the probability density weights m/z regions according to their size. If the probability density favors abundant ions from the training set too heavily, even for large N, pseudo-counts are added to the m/z regions. As the pseudo-counts increase, the m/z regions are sampled according to their size and all information about the m/z values of the abundant ions in the training spectra is lost.

Once the empirical distribution of m/z values and intensities from the training spectra are established, random spectra are generated by choosing 10 peaks according to the discrete intensity and m/z region probability densities. Three peaks are assigned intensity 100%, with the rest drawing their intensity independently from

the empirical intensity distribution. Each of the peaks draws their m/z value from the empirical m/z value distribution independently of each other, and their intensity values.

The parameters of this random spectrum generation technique serve to favor, or discount, the selection of abundant training spectra m/z values. Initially set the pseudo-counts are set to zero and the parameter N is adjusted until the distribution of the HMMatch scores of random spectra matches the distribution of the HMMatch scores of HC-Other spectra. Pseudo-counts are used if no value of N is sufficient.

3.2.6.3 Score Distribution and p-value Estimation

Experiments results show that the normal distribution, with appropriately chosen mean and variance, is a good fit to the empirical shape of the HMMatch scores from random spectra. The mean and variance of HMMatch scores of 1000 random spectra are easily computed, and are used to transform HMMatch scores to normal distribution z-scores to estimate p-values.

3.2.7 Implementation

HMMatch was written in C++. It is implemented on top of LAMP_HMM v. 0.9. LAMP_HMM is a general implemtation of HMM algorithm. It is implemented by Daniel DeMenthon and Marc Vuilleumier. The code was downloaded from `http://www.cfar.umd.edu/~daniel/Site_2/Code.html`. It allows observations to be vectors and the observation probabilities to be modeled by histograms

or Gaussians. It also provides both Baum-Welch method and segmental k-means method for training. The other implementation properties not related to HMMatch are listed on the software website.

The mass spectra data used for training and testing HMMatch are processed and provided by Professor Nathan Edwards. The data selection approach is described in section 3.2.3.

3.2.8 Experimental Results

3.2.8.1 Experimental Data

For our experiements, we selected eight peptide/charge state combinations of mass spectra from the relational database of identified spectra. The peptides, their m/z value, and the number of spectra in each category is shown in Table 3.1. Peptides were selected from those with the most high-confidence identifications in the relational database.

3.2.8.2 Training

After training the HMMatch models using the Baum-Welch algorithm the performance of each model on each spectrum class is evaluated. Given the relatively small number of training examples for each peptide, special attention must be paid to the possibility of over-fitting the model. This issue is carefully considered in the hidden Markov model design, particularly in the spectrum discretization and emission probability independence assumption, and in the initialization of intensity

Peptide							
m/z	z	HC-Train	HC-Test	LC	HC-Other	LC-Other	Unknown
DFLAGGVAAAISK							
610.34	2	36	23	59	48	879	22481
DFLAGGIAAAISK							
617.35	2	33	36	14	113	804	5001
DLATVYVDVLK							
618.35	2	36	40	61	159	772	4615
AVMDDFAAFVEK							
671.82	2	110	82	143	205	820	5543
LNDLEDALQQAK							
679.35	2	28	28	45	169	741	7591
AVM*DDFAAFVEK							
679.82	2	56	50	52	116	723	7585
SHCIAEVENDEMPADLPSLAADFVESK							
992.12	3	27	25	140	162	1317	6208
SHCIAEVENDEM*PADLPSLAADFVESK							
997.45	3	40	40	125	128	1192	6074

Table 3.1: Model peptides and spectral data set sizes. Cysteines alkylated with iodoacetamide. * indicates oxidized methionine.

emission probability distributions. Plotting the distribution of HMMatch scores (Figure 3.5) for the HC-Train and HC-Test spectra shows that there is no evidence of over-fitting. In each case, the distribution of training scores match the distribution of testing scores. Furthermore, it is observed that there is excellent separation between the HMMatch scores of the spectra with high-confidence identifications to the model peptide (HC-Test) compared with that of other peptides (HC-Other). This implies that HMMatch has similar specificity as X!Tandem with respect to spectra with high-confidence identifications. Because the distribution of viterbi scores and forward scores are very similar, only results for forward scores are shown here. This is mainly because of the proposed data pre-processing and model building algorithm, which give very sparse transition matrix with most non-zero elements lie along the diagonal.

The time to train each HMMatch model varies depending on the number of states, the number of training spectra, and the number of Baum-Welch iterations required for convergence. Table 3.2 shows these statistics for each model. Most of the HMMatch models were trained in less than 10 seconds, with the longest training time less than one minute.

Computation of the HMMatch score (for both viterbi score and forward score) is also quite fast. Table 3.3 shows the time to compute HMMatch scores for each peptide's spectra. Viterbi distance evaluation time depends primarily on the number of states in the hidden Markov model. While more time-consuming than the sequence and dot-product based search engines, the time to compute HMMatch scores is not prohibitive, even for very large spectral data sets, despite the use of a

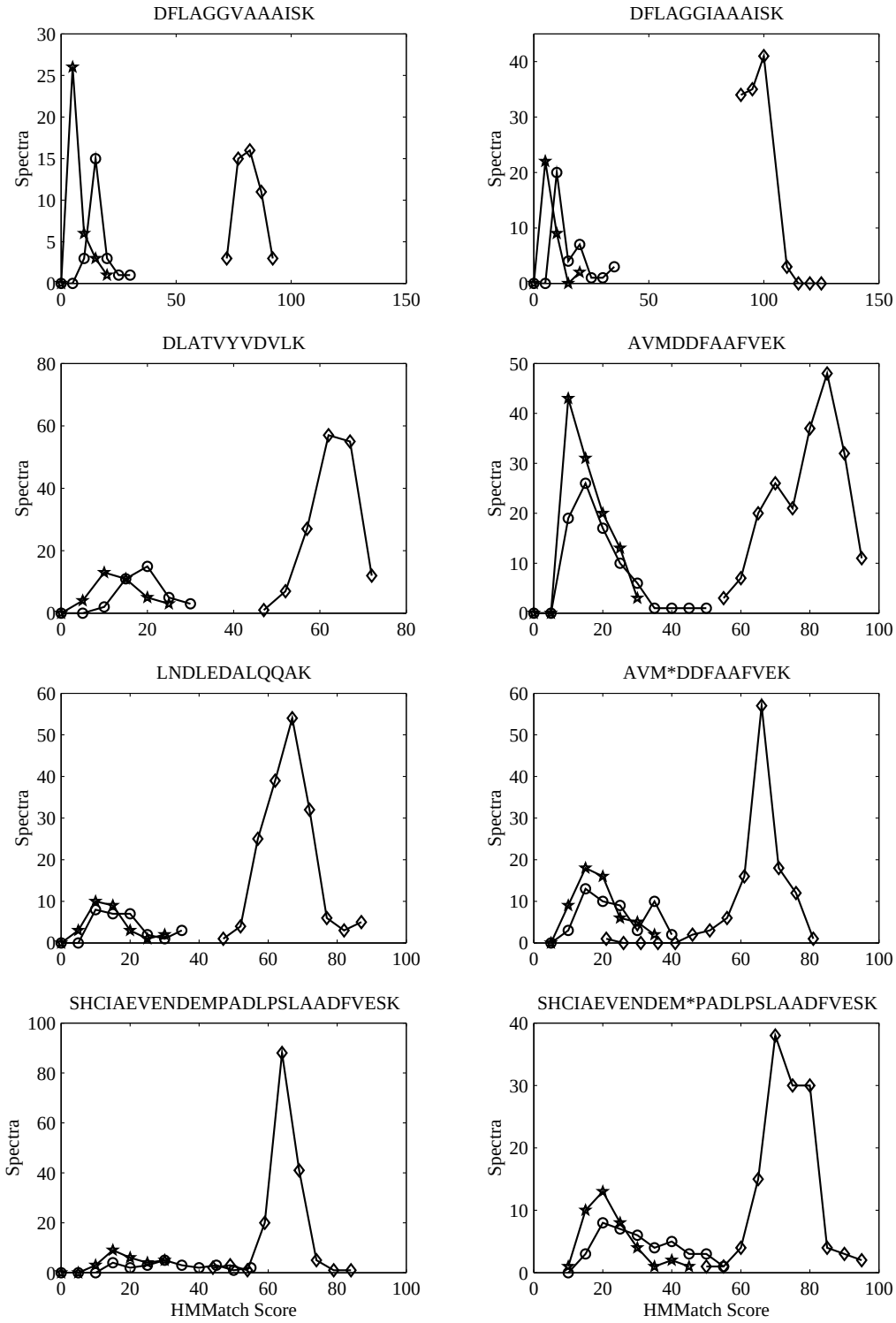

Figure 3.5: HMMatch forward scores for spectra with high-confidence identifications (X!Tandem E-value $< 10^{-4}$). High-Confidence Train (HC-Train) $\star$; High-Confidence Test (HC-Test) $\circ$; High-Confidence Other (HC-Other) $\diamond$.

Peptide	Time (s)	Iterations	States	Spectra
DFLAGGVAAAISK	6	2	81	36
DFLAGGIAAAISK	5	2	81	33
DLATVYVDVLK	4	2	69	36
AVMDDFAAFVEK	16	2	75	110
LNDLEDALQQAK	3	2	75	28
AVM*DDFAAFVEK	8	2	75	56
SHCIAEVENDEMPADLPSLAADFVESK	16	2	120	27
SHCIAEVENDEM*PADLPSLAADFVESK	44	2	141	40

Table 3.2: Time (in seconds) for HMMatch training.

generic hidden Markov model codebase that does not take advantage of our model's sparsity.

3.2.8.3 Basic Performance

The sensitivity of HMMatch is evaluated by plotting the distribution of HMMatch score p-values for spectra with weak or no identifications. Figure 3.6 give the p-values for Forward scores. Because Viterbi scores and Forward scores have very similar distribution, their p-values have similar distribution too, here only p-values for Forward scores are shown. Performance wise, a large proportion of the spectra with weak identifications to the model peptide (LC) have very small p-values, demonstrating higher confidence than X!Tandem for these peptides. A few weak identifications to other peptides (LC-Other) have quite significant HMMatch scores which suggests the weak X!Tandem identifications are incorrect. Lastly, a considerable fraction of the spectra with no identification with E-value ≤ 1 (Unknown) have significant HMMatch scores. A number of thes cases are manually examined and discussed in the following paragraphs. In each case it is found that spectra with significant HMMatch scores are an excellent match to high confidence spectra from the model peptide.

The performance of HMMatch is tested as the number of training spectra is reduced. HMMatch models for the peptides LNDLEDALQQAK and DFLAGGIAAAISK are constructed using training sets consisting of 5, 10, 20, and 40 randomly selected HC spectra. For peptide LNDLEDALQQAK, the distribution of

Peptide	Time (s)	Spectra	Spectra/s
DFLAGGVAAAISK	586	23526	36.47
DFLAGGIAAAISK	143	6001	41.11
DLATVYVDVLK	90	5683	62.46
AVMDDFAAFVEK	137	6903	48.96
LNDLEDALQQAK	175	8602	48.87
AVM*DDFAAFVEK	172	8582	48.49
SHCIAEVENDEMPADLPSLAADFVESK	629	7879	13.33
SHCIAEVENDEM*PADLPSLAADFVESK	919	7599	8.45

Table 3.3: Time (in seconds) to compute HMMatch scores.

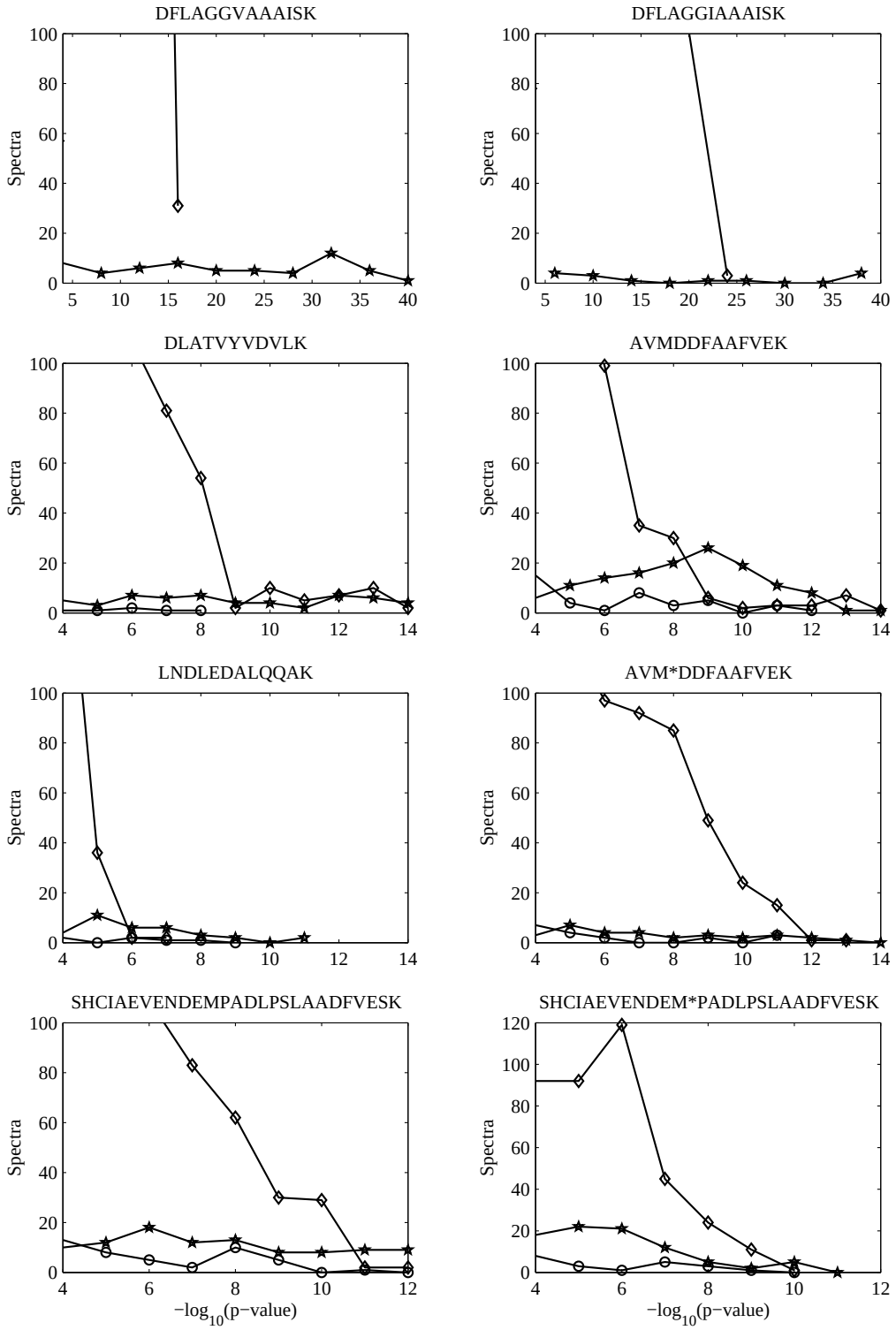

Figure 3.6: HMMatch forward score based p-values of low-confidence and unknown spectra (X!Tandem E-value $> 10^{-4}$). Low-Confidence (LC) ⋆; Low-Confidence Other (LC-Other) ○; Unknown ◇.

HMMatch scores for HC-Test and HC-Other overlap a little for 5 training spectra, but are completely separated for 10, 20, and 40 training spectra. For peptide DFLAGGIAAAISK, the HC-Test and HC-Other scores are well separated for all training set sizes from 5 to 40. As with all machine-learning techniques, HMMatch training is more effective as the number of training examples increases, however, these experiments demonstrate that good performance is possible even with a relatively small number of training spectra per peptide.

3.2.8.4 Comparative Performance

Because of the absence of a sufficiently rich data set with known correct peptide identifications, it is not easy to establish performance in terms of sensitivity and specificity. To provide reasonable performance results, HMMatch is compared with sequence database search engines X!Tandem and Mascot, and spectral matching search engine MS Search from NIST with consensus spectra from the NIST library of peptide fragmentation spectra. Precision-recall statistics are used to compare the tools' peptide identification scores. Some of the cases are manually examined to show that HMMatch is able to confidently identify some spectra, while the other tools cannot.

Performance Comparison. Each tool orders the spectra in each data set according to some spectrum-peptide match score. For X!Tandem and Mascot, this is the *E*-value, for NIST's MS Search this is the match factor, and for HMMatch this is the HMMatch score described above. Given a reference labeling as positive or negative

with respect to the model peptide, it can be evaluated that, with respect to any spectrum-peptide match score threshold, the extent to which the partition of the spectra using this threshold is consistent with the reference labels. The spectrum-peptide match score and threshold and the reference labels partition the spectra into true-positive (TP), false-positive (FP), false-negative (FN), and true-negative (TN) sets. Precision, then, is defined as $|TP|/(|TP| + |FP|)$, while recall is defined as $|TP|/(|TP| + |FN|)$. Perfect correspondence with the reference labels, for a particular match score and threshold, results in 100% precision and 100% recall. The precision-recall curve, which plots the precision-recall statistics for all thresholds, captures the extent to which the ranking of spectra by some match score is consistent with the reference partition of the spectra into positive and negative examples. Perfect correspondence with the reference labels results in a square precision-recall curve.

Each tool is compared to the others by using each tool and some spectrum-peptide score threshold, in turn, to construct *synthetic* reference labels. As such, these precision-recall statistics and curves do not represent true measures of sensitivity and specificity, instead they capture the extent that each tool's spectrum-peptide match score ranks the spectra in an order that is consistent with the synthetic reference. Figure 3.7 show a complete set of precision-recall curves for the synthetic reference based on X!Tandem E-value and significance threshold 0.01. To summarize the comparative behavior more comprehensively, table 3.4 show the % recall at 99% precision for each tool averaged across the eight data sets, with respect to synthetic reference labels derived from each tool's spectrum-peptide match score.

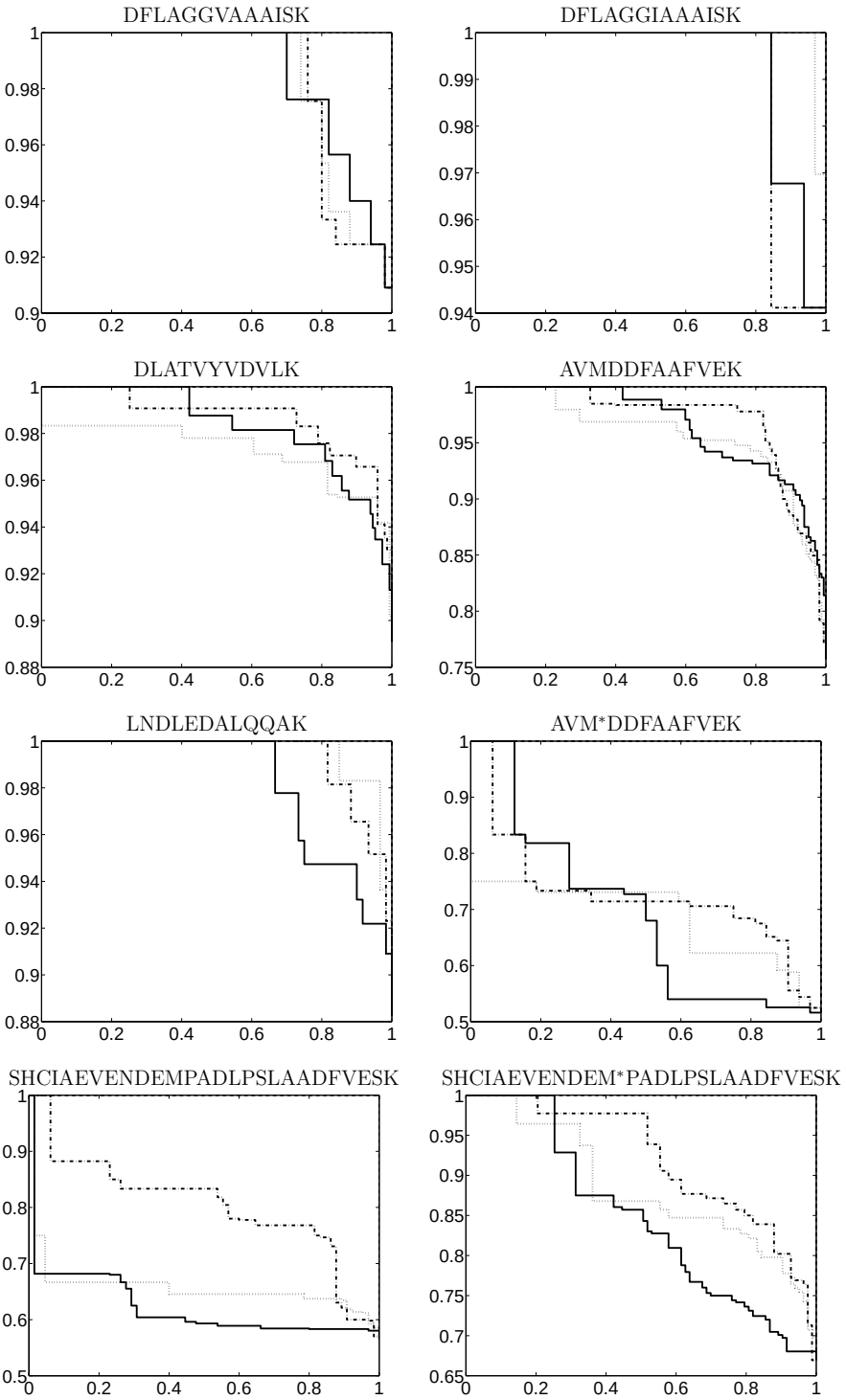

Figure 3.7: Precision-recall curves for Mascot (- - -), MS Search (·····), and HMMatch (——) for synthetic reference labels defined by X!Tandem E-value threshold 0.01. Training spectra and spectra with no X!Tandem E-value for the model peptide are excluded.

Reference	**Threshold**	**Pos./Neg.**	**X!Tandem**	**Mascot**	**MS Search**	**HMM**
X!Tandem	0.1	93 / 12	-	59%	54%	53%
(E-value)	0.05	90 / 15	-	56%	54%	54%
	0.01	79 / 26	-	48%	37%	43%
	0.001	61 / 44	-	38%	14%	32%
Mascot	0.1	38 / 67	38%	-	33%	40%
(E-value)	0.05	34 / 71	42%	-	35%	34%
	0.01	26 / 79	45%	-	26%	33%
MS Search	0.8	52 / 63	46%	61%	-	60%
(Match Factor)	0.9	62 / 53	36%	55%	-	49%
	0.95	72 / 43	32%	42%	-	46%

Table 3.4: Average % recall at 99% precision for 8 model-peptide spectrum data sets with respect to various synthetic reference labels. Training spectra and spectra with no reference score excluded.

The precision-recall curves and statistics show only a moderately good correspondence between the ranking imposed by *any* pair of tools. Table 3.4 shows that for high precision identification, the HMMatch score is more like X!Tandem and Mascot than MS Search, particularly for the smaller E-value synthetic label thresholds. For spectral matching synthetic labels from MS Search, HMMatch shows a similar degree of correspondence as Mascot, with X!Tandem showing quite a bit less. The precision-recall curves in Figure 3.7 show there is no tool that is uniformly more consistent with X!Tandem than the others. For four of the eight data sets, the HMMatch score recalls the most spectra at 100% agreement with X!Tandem, Mascot and MS Search recall the most for two each of the remaining data sets. Peptide SHCIAEVENDEMPADLPSLAADFVESK shows considerable disagreement between the search engine tools and the spectral matching based tools, with strong agreement between the spectral matching based tools. On the other hand, peptide AVM*DDFAAFVEK shows considerable difference between the spectra matching based tools, as compared to X!Tandem. On the strength of the precision-recall curves and statistics, it is not hard to conclude that HMMatch shows a similar level of concordance with the other tools as they do to each other. It also shows that HMMatch does not seem to be overtrained or biased towards the X!Tandem identifications, despite the use of these identifications in HMMatch training.

Case Study. Our data sets contain many spectra with highly significant HMMatch scores but with poor scores from X!Tandem and Mascot. HMMatch confidently identifies, with HMMatch score p-value $< 10^{-5}$, 3537 spectra with both Mascot and

X!Tandem E-values greater than 0.05. By way of comparison, NIST's MS Search confidently identifies, with match factor threshold 0.9, only 673 spectra with both Mascot and X!Tandem E-values greater than 0.05. A number of these spectra are manually examined to determine whether or not HMMatch was likely correct and to try to explain the poor scores from other tools.

A considerable proportion ($\tilde{3}50$) of the spectra were correctly identified by HMMatch and misidentified by the search engines due to peptide candidate selection issues. It was observed that there were a large number of misidentified spectra with precursor molecular weight outside of the allowed search engine precursor tolerance of 2 Da. The spectral matching techniques also use a precursor tolerance of 2 Da, but it is applied to the m/z value of the precursor. When the search engine parameter is increased to 4 Da (6 Da) for charge 2 (charge 3) peptides, these spectra are usually confidently identified as the corresponding model-peptide by X!Tandem and Mascot. So, while these spectra are confidently identified by HMMatch, they are not a good demonstration of the strength of HMMatch scores as compared to the peptide sequence based scoring. However, it does suggest that the usual 2 Da precursor tolerance used for sequence searching is perhaps too aggressive. Nevertheless, increasing the precursor tolerance increases Mascot and X!Tandem E-values by a similar factor, further reducing their sensitivity.

Another class of spectra correctly assigned by HMMatch, but with poor X!Tandem and Mascot scores, are noisy spectra with many extraneous peaks, as in the fragmentation spectrum of peptide DLATVYVDVLK in Figure 3.8(a). This spectrum, confidently identified by HMMatch with p-value 7.341×10^{-12} has Mascot E-value

1.07 and X!Tandem E-value 0.0013. While X!Tandem's E-value is less than typical significance thresholds, the E-value does not indicate the same degree of confidence as HMMatch. The sequence database search engines have trouble with these spectra as the extra peaks tend to match fragment ions whether or not the correct peptide is being evaluated. The MS Search spectral matching search engine computes a match factor of 0.844 for this spectrum-peptide combination, well below match factors for other high-confidence peptide identifications. The dot-product based match factor is affected by noisy spectra, as documented by [MSS05]. It is believed that the aggressive peak selection strategy used for spectrum normalization makes the HMMatch robust with respect to the presence of extraneous noise peaks, without compromising identification performance.

The fragmentation spectrum, also of peptide DLATVYVDVLK, in Figure 3.8(b) represents another interesting failed identification by Mascot and X!Tandem that is correctly identified by HMMatch. The HMMatch score has significance 1.738×10^{-12}, while MS Search computes a match factor of 0.994. Mascot, on the other hand, computes an E-value of 5.7 and X!Tandem an E-value of 0.16, neither of which is statistically significant. Close examination of the search engine results revealed that the spectrum received poor scores due to the fragment ion mass tolerance parameter. The X!Tandem and Mascot searches were conducted using a fragment ion match tolerance of 0.4 Da, a relatively conservative fragment tolerance appropriate for the ion trap spectra that makes up the majority of publicly available MS/MS spectra. However, for the spectrum of Figure 3.8(b), b_4 and y_3 ions, amongst others, were observed more than 0.4 Da from their theoretical value, which was sufficient

to drive down the search engine scores. Increasing the fragment tolerance increases Mascot and X!Tandem E-values, further reducing their sensitivity. HMMatch uses a fragment tolerance of 0.5 Da, and matched b_4, but didn't even use the peak that matched y_3 as it was outside of the 10 most intense peaks. The MS Search match factor was computed using the default bin-size of 0.8 Da, so the measurement error apparent in these fragment ions did not affect its score. We believe that the use of peak intensity by HMMatch makes it possible to use a larger fragment ion tolerance and more aggressive peak selection, without sacrificing identification performance.

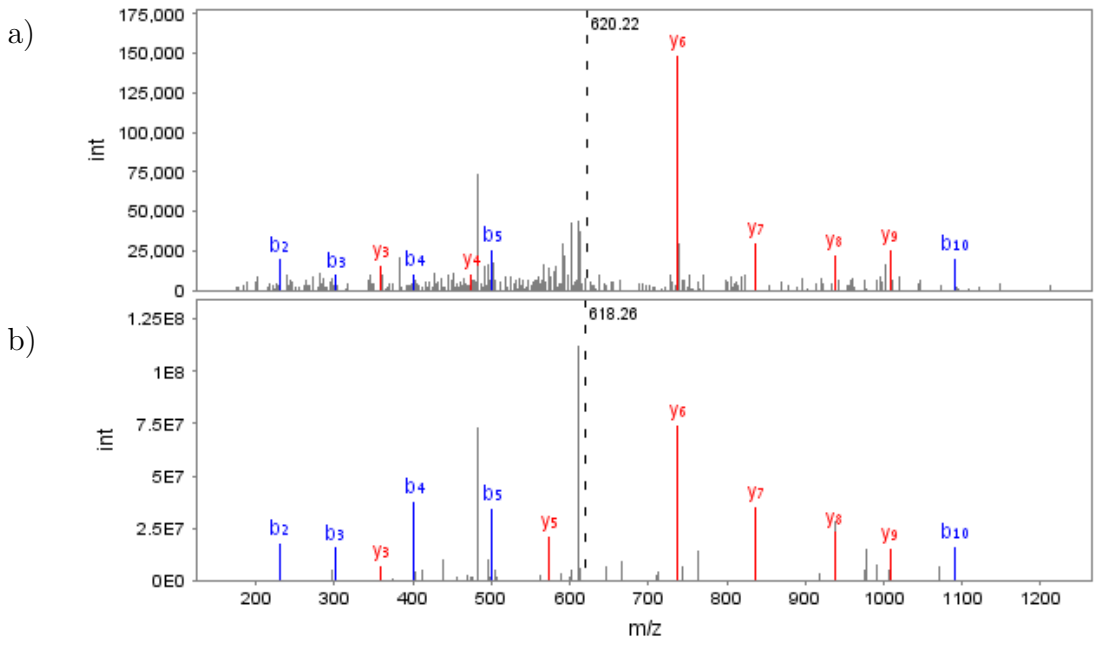

Figure 3.8: Case study fragmentation spectra of peptide DLATVYVDVLK. (a) Mascot E-value: 1.07, X!Tandem E-value: 0.0013, MS Search match factor: 0.844, HMMatch p-value: 7.341×10^{-12}; (b) Mascot E-value: 5.7, X!Tandem E-value: 0.16, MS Search match factor: 0.994, HMMatch p-value: 1.738×10^{-12}.

3.2.9 Model Extrapolation

Traditional spectral matching, which uses dot-product based similarity scores, essentially treats the reference spectra as bitmapped images with no semantic content. The use of artificial consensus spectra, which implicitly encodes semantics by retaining only frequently observed or expected peaks and permits asymmetric variants of the dot-product based similarity measures, is the approach adopted by NIST for its GC/MS [SS94] and peptide fragmentation spectral libraries, as well as the MS Search spectral matching search engine.

The construction of a probability model, such as HMMatch, to abstract and semantically summarize the behavior of a set of peptide fragmentation spectra makes it possible to *extrapolate* the model to spectra from other, related, peptides. Two peptides that differ by a single amino acid or by the addition or removal of a post-translational modification will, in many cases, have similar normalized intensities, once an appropriate offset is applied to some of the ions. For example, the peptides DFLAGGVAAAISK and DFLAGGIAAAISK differ by a Val to Ile substitution, a mass shift of +14.02 Da, which affects the m/z value of all the fragment ions that contain the changed residue, including $b_7, \ldots, b_{12}$ and $y_7, \ldots, y_{12}$, and of course the precursor. Other than this mass shift, however, the peptide fragmentation spectra of these peptides are very similar. Figure 3.9 shows the spectral box-plot of HC-Train spectra, defined as for Figure 3.2, for each of these peptides. This figure, which plots the distribution of intensities in each of the discrete m/z regions of $\mathcal{R}$, abstracts away the shift introduced by the amino acid substitution and makes the

similarity apparent.

Given the semantic abstraction of our trained HMMatch model, it is straightforward to compute the HMMatch scores of spectra of one peptide using the HMMatch model of the other. Our set of spectra includes three such pairs: DFLAGGVAAAISK and DFLAGGIAAAISK, AVMDDFAAFVEK and AVM*DDFAAFVEK; and SHCIAEVENDEMPADLPSLAADFVESK and SHCIAEVENDEM*PADLPSLAADFVESK. The last two pairs differ by an oxidized methionine (+15.99), indicated by *, while the first pair has an amino acid substitution, as already described. For each peptide, we compute the p-value of the HMMatch score of HC-Test spectra using both the original HMMatch model and the extrapolated model of its twin. Figure 3.10 plots the original Viterbi score based p-value vs the extrapolated Viterbi score based p-value for each of the six paired peptides. The line $y = x$ is added to provide a visual aid for estimating the number of HC-Test spectra with increased or decreased significance.

It is observed that while the p-values are somewhat scattered, most spectra with statistically significant HMMatch scores computed using the correct model are still statistically significant with respect to the extrapolated twin peptide's HMMatch model. In fact, a good number of the HC-Test spectra are *more* significant (lie above the $y = x$ line) when scored using the twin's model. The success of this model extrapolation suggests that HMMatch models may be created and used to identify peptides from a specific isoform or modification, even when their spectra have not previously been identified by other tools. Similarly, it may be possible to train a single HMMatch model using fragmentation spectra from two related

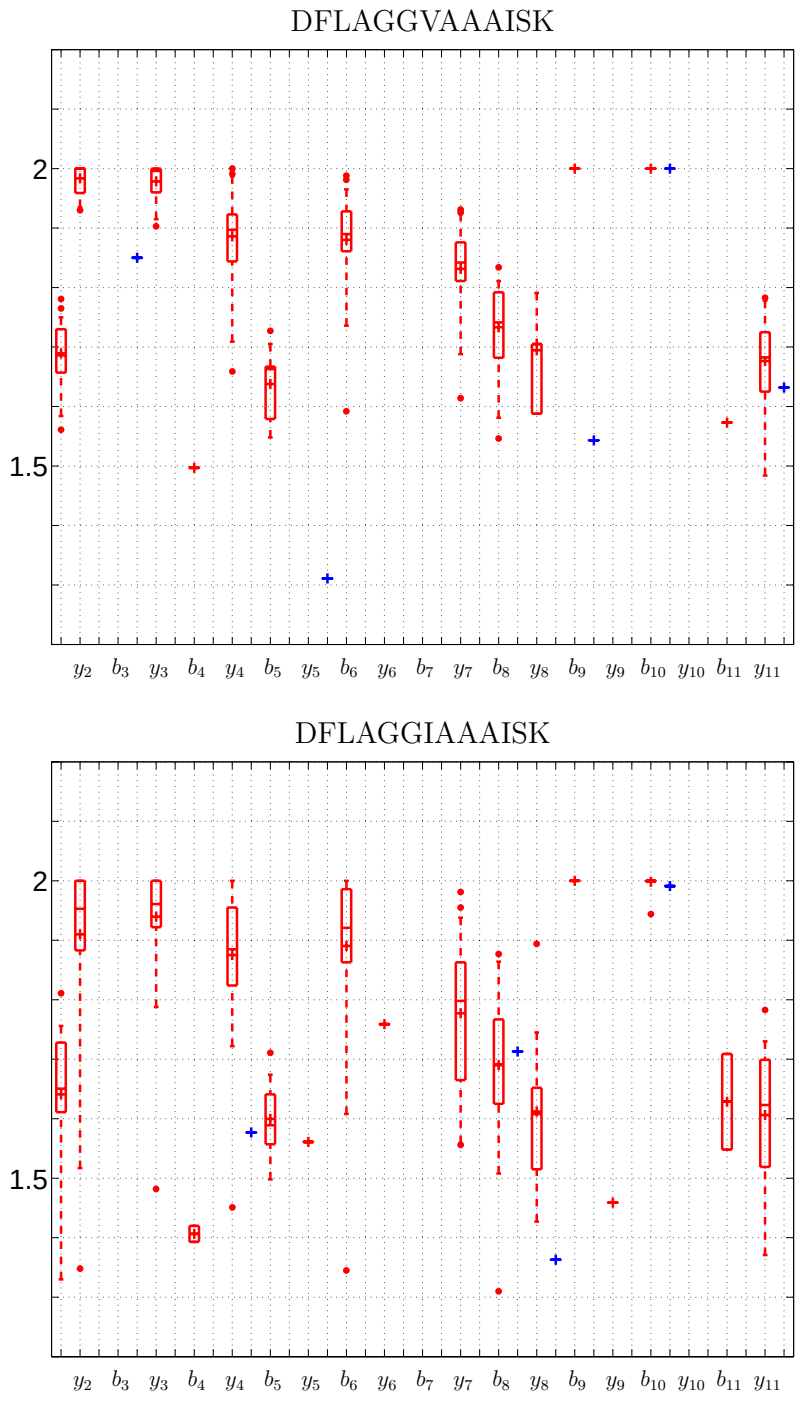

Figure 3.9: Normalized intensity boxplots for peaks in each m/z value region from HC-Train spectra of DFLAGGVAAAISK and DFLAGGIAAAISK.

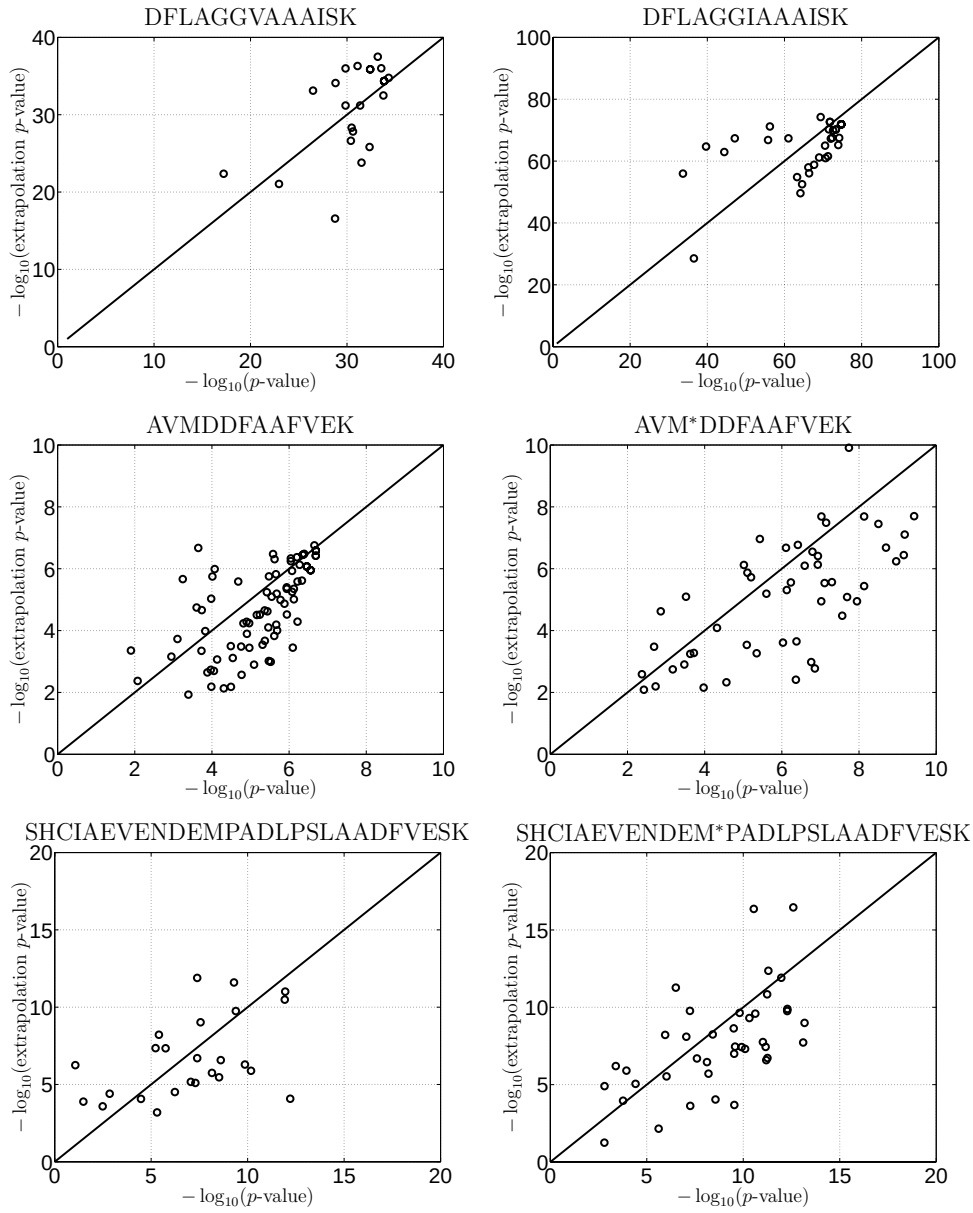

Figure 3.10: Comparison of p-values of HC-Test spectra scored with the peptide's HMMatch model and the extrapolated HMMatch model of its related "twin" peptide.

peptides, useful when there are too few high-confidence training spectra for each individual peptide.

While many related peptides have similar (up to mass shift) spectra, this is clearly not true for all amino acid substitutions (in particular, the insertion of a basic residue) and post-translational modifications. While it is not necessarily be able to predict which related peptides will have similar spectra, the statistical significance computation ensures that extrapolated model assignments are not accepted when they are unlikely to be correct.

A larger problem for HMMatch is the linear, ordered, structure of the expected ion hidden states. If the masses of expected or common fragments ions after the mass shift is applied are no longer correctly ordered, then it is unclear how the hidden Markov model's transition probabilities should be adjusted to compensate. For each of the peptide pairs above, the mass shift is small enough that this is not an issue.

3.2.10 Conclusion

HMMatch is a novel approach for spectral matching based peptide identification. It uses a hidden Markov model to summarize the statistical variation and consensus in a peptide's fragmentation spectra and applys this model to the identification of unassigned spectra.

The experiments results indicate that HMMatch has good specificity and superior sensitivity, compared to sequence database search engines such as X!Tandem.

The HMMatch design achieves good results from relatively few training spectra, is fast to train, and can evaluate many spectra per second. A statistical significance model permits HMMatch scores to be compared with each other (and with other peptide identification tools) on a unified scale. HMMatch shows a similar degree of concordance with X!Tandem, Mascot, and NIST's MS Search, as they do with each other. This suggests that each tool can assign peptides to spectra that the others miss. Finally, it is shown that it is possible to extrapolate HMMatch models beyond a single peptide's training spectra to the spectra of related peptides, thus expand the application of spectral matching techniques beyond the set of peptides previously observed.

As the popularity of protein characterization by tandem mass spectrometry grows and the public repositories of peptide fragmentation spectra to increase in size, covering a larger proportion of more organisms' proteomes with more examples of mass spectra from a variety of instruments, the hidden Markov model approach to spectral matching will become increasingly useful in the analysis of peptide fragmentation spectra.

3.3 PepArML: An Unsupervised, Model-Free, Combining Peptide Identification Arbiter for Tandem Mass Spectra Via Machine Learning

3.3.1 Related Work

As mass spectrometry techniques advance, many database search algorithms and tools are being developed to provide more precise peptide identification results. However, current algorithms for peptide identification have much room for improvement. Researchers have proposed many different approaches for enhancing existing proteomics tools. A variety of techniques have been applied to protein sequence database search engine outputs in order to improve the reliability of peptide identification. In general, these techniques treat the search engines as "black boxes", and try to do a better job at distinguishing correct from incorrect peptide identifications by directly processing search engines' peptide identification results. These black-box tools employ one or more of the following techniques: combining or merging of search engine results, supervised and semi-supervised machine learning scoring and prediction, statistical significance re-estimation.

3.3.1.1 Combining Multiple Search Engines

One of the important techniques that researchers proposed is to combine results from multiple peptide identification tools. These result combiners [RMAM$^+$04, STN08, HKB$^+$07] rely on peptide identification results from multiple search engines

to improve sensitivity and specificity, with the assumption that search engine agreement increases the likelihood that the identification is correct.

Higgs et al. [HKB+07] uses blackbox techniques to combine peptide identification results from multiple search engines (SEQUEST and X!Tandem). They use reverse database search results as null models to re-estimate the statistical significance of the peptide identifications.

Scaffold [STN08] is a commercial tool from Proteome Software that uses statistical algorithms to combine results from multiple search engines to calculate the probability that proteins are actually in a biological sample. Statistical significance re-estimation techniques are used to normalize and compare scores from multiple search engines. Scaffold also provides an interface for displaying proteins identified in a tandem mass spectrometry experiment.

3.3.1.2 Applying Machine Learning To Improve Scoring

Supervised machine learning techniques [KNKA02, UZQA06, HKB+07, ALPN03, BBIK04] have been applied to existing and novel features derived from search engines' results. These derived score functions demonstrate better sensitivity and specificity for specific data sets.

PeptideProphet [KNKA02] was designed to re-estimate the statistical significance of SEQUEST's peptide identification results. It uses expectation maximization algorithm to learn to distinguish correct from incorrect database search results. The author claims this approach also applies to other database search based peptide

identification algorithm. PeptideProphet first uses a supervised training phase to establish the optimal linear weighting of the various SEQUEST peptide identification properties and scores. Then it uses an unsupervised empirical fit of a bimodal mixture model to the discriminant scores to establish the likelihood of correct peptide identifications.

Ulintz et al. [UZQA06] try to classify the database search engines' peptide identification results by applying machine learning algorithms (Boosting and Random Forest) on peptide identification results. They not only directly use the peptide identification results given by the search engines, but also design new metrics to be incorporated in the feature vector correspondent to each peptide identification result. The machine learning framework show significant improvement over original search engines peptide identification results.

Anderson et al. [ALPN03] use machine learning algorithms, support vector machine, and 13 carefully constructed metrics as features to distinguish correct from incorrect peptide identifications output by SEQUEST. The results show machine learning based approach improves sensitivity compared with the original SEQUEST scores.

Baczek et al. [BBIK04] use artificial neural network (ANN) to re-analyze SEQUEST's peptide identification results based on a series of calculated peptide identification metrics. The results show ANN can be used to automate peptide identification results' classification even though they applied strict restriction when selecting the training and testing data.

Supervised machine learning techniques are also often combined with statisti-

cal significance re-estimation techniques [KNKA02, HKB+07] to normalize the class prediction probabilities.

3.3.1.3 Re-estimating Statistical Significance

There have been several techniques to estimate statistical significance of peptide identification results. The unsupervised E-M algorithm phase of PeptideProphet [KNKA02] uses empirical properties of the underlying search engine results as a null-model to re-estimate the statistical significance or likelihood associated with peptide identification results. Its unsupervised significance re-estimation algorithm helps to make its probabilities quite robust, as long as the underlying linear discriminant model generalizes well enough to distinguish correct and incorrect identifications.

Other tools such as Qscore [MYL02] use a theoretical model to re-estimate the significance of peptide identification results. Qscore was one of the first algorithms that tries to use blackbox techniques to re-classify database search engines' peptide identification results. The scoring system incorporated more metrics about peptide matching quality other than original SEQUEST scores. The results show that the re-estimated statistical significance score performs better for distinguishing correct from incorrect peptide identifications.

A current popular strategy is to use a decoy database [EHFG05a, EHFG05b, HKB+07] to re-estimate the statistical significance of search engine outputs. A decoy protein sequence database is constructed using either reversed or shuffled protein

sequences. Peptide sequences in the decoy databases are assumed to be non-existent and will not match experimental mass spectra. Any peptide identifications from sequences in the decoy databases can thus be assumed to be false. By measuring the proportion of peptide identifications that are generated by the decoy database at different score thresholds, the statistical significance of search engine scores may be re-estimated.

Elias et al. [EHFG05a, EHFG05b] mentioned several approaches for estimating false discovery rates in their papers. They provided detailed descriptions of different approaches in the supplementary section of their papers. We use the method they propose for calculating false discovery rates since they seem to be the research group performing the most extensive studies on estimating false discovery rates.

3.3.2 Motivation

A major concern with supervised machine learning techniques is the question of how to generalize the trained machine learning model to peptide identification results from a different instrument or search engine. One solution is to use unsupervised or semi-supervised training. This approach makes it possible for the training to be applied on the fly to each new data set. The procedure can also be adapted to the particular features of a given set of results as needed. Semi-supervised machine learning has recently emerged [CN08, KCW+07] as a solution to the concerns about the generality of supervised machine learning. This type of approach uses decoy database hits as known false peptide identifications to guide the machine learning

approach.

A second concern with the black-box output re-analysis techniques is that they require the correct (and incorrect) peptide identifications follow some underlying mathematical model. Sometimes these model assumptions are explicit, especially when using LDA [KNKA02] and SVM [ALPN03] machine learning techniques. While in other cases they are implicit, such as the assumption that incorrect peptide identifications from different search engines are uncorrelated [RMAM+04, STN08, HKB+07].

To ally these concerns, we propose and test PepArML, a *model-free, result combining, unsupervised machine-learning approach* to improving the sensitivity of peptide identifications. Because PepArML does not rely on a fixed model or require pre-labeled training data, it can be adapted to individual data sets on the fly automatically. Experimental results indicate it outperforms each of the described basic techniques.

3.3.3 Performance Metric Definitions

Throughout this section, a number of standard performance metrics are used to evaluate the peptide identification results from different algorithms and database search engines. For brevity we refer to an individual mass spectrum to peptide identification generated by a search engine as a *peptide ID*.

3.3.3.1 Definitions

We define the *ground truth* as the correctness of each peptide ID. We can determine ground truth only for a synthetic protein mixture where we know the identity of each protein in the mix. A peptide ID is considered to be true if the spectrum is assigned a peptide from one of the proteins known to be in the synthetic protein mix. A peptide ID is considered to be false if the spectrum is assigned to a peptide from any other protein in the database. For the selected peptide IDs, the number of true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) is counted with respect to the known ground truth.

Sensitivity, defined as TP/(TP+FN), is also known as true positive rate (TPR). It measures the percentage of true cases that are correctly classified (identified). Specificity is defined as TN/(FP+TN). It measures the percentage of false cases that are correctly classified. The false positive rate (FPR) is defined as (1−specificity).

False discovery rate (FDR), defined as FP/(FP+TP), measures the percentage of false positives among classified true cases. Alternatively, FDR can be viewed as the expected proportion of false positives among the declared significant results. Notice that FDR and FPR are two different values.

FDR is a method commonly used in multiple hypothesis testing in order to correct statistical significance for multiple comparisons. It is needed since E-values reported for each peptide ID only reflect the statistical significance of individual predictions. When combining results for thousands of peptide IDs, FDR provides a more accurate prediction of the statistical significance of the entire collection of

predictions.

3.3.3.2 Calculating FDR and Estimated FDR (eFDR)

To compute FDR requires knowledge of ground truth for each data set, which is not possible for experiments with unknown proteins. Fortunately, decoy databases (described in Section 3.3.1.3) may be used to compute an *estimated false discovery rate* (estimated FDR, or eFDR) for each peptide identification algorithm [EHFG05a], by considering all peptide IDs from sequences in the decoy database to be false positives. When calculating eFDR using decoy protein sequence databases, the original (true) protein sequences are referred to as the *forward database*, and searching the original database is referred to as a *forward database search*.

We use #F to represent the number of peptide IDs from the forward database search. #Ft then represents the number of true positives in forward database search, and #Ff represents the number of false positives in forward database search. Since #F = #Ft + #Ff, FDR can calculated as:

FDR = #Ff/#F = #Ff/(#Ft + #Ff)

Similarly, we use #R to represent the number of peptide IDs from the decoy database. It turns out that #R can also be used to estimate the number of false positives in forward database search (#Ff), based on assumption that the random matches appear with equal frequency in searches against both the forward and decoy database sequences.

Based on this assumption, FDR can be represented as either:

$$FDR = \#R/\#F$$

or

$$FDR' = 2\#R'/(\#F' + \#R')$$

if the forward and decoy database (of equal size) are concatenated. #F' and #R' are number of peptides IDs from forward and decoy database when the forward and decoy databases are concatenated, so their values can be smaller than #F and #R. In practice, no method for calculating estimated FDR has been proven optimal. Theoretically, concatenated database and using #R/#F provides better FDR estimations. But in real experiments, especially when results with FDR $\leq$ 0.2 are most important, using either concatenated or separate forward & decoy databases (and either #R/#F or 2#R/(#F+#R)) seem to provide similar results. However, since scores (E-values) from different search engines need to be calibrated, in the thesis we use the method suggested by Elias *et al.* [EHFG05a] to calculate estimated FDR as 2#R'/(#F' + #R').

3.3.3.3 Experimental Metrics

To the experiment evaluation section, we use a number of different metrics to evaluate the performance of each peptide identification algorithm:

Receiver Operating Characteristic (ROC) curves. ROC are plots of the sensitivity vs. (1–specificity), which equals to the true positive rate vs. the false positive rate, for all possible threshold values. ROC curves provide a global view of the sensitivity/specificity tradeoff of a particular predictor,and evaluate the predictor's ability

to separate true from false peptide IDs.

Area under ROC (AUROC). AUROC is a single-value summary of a ROC curve, where higher values are desirable. A predictor that separates true and false peptide IDs perfectly will have a square ROC curve and AUROC of 1.

Sensitivity for given FDR. The sensitivity of each predictor is evaluated at score thresholds for different values of FDR or estimated FDR. These graphs are similar to ROC curves, but somewhat more practical since eFDR may be computed without knowing actual ground truth (which is needed for computing ROC and true FDR). This metric can be used to evaluate predictor performance on a data set derived from real biological samples, even though estimated FDR may not approximate true FDR very well.

3.3.4 Heuristics For Combining Search Engine Results

To test whether machine learning is necessary to achieve good performance, we designed several simple heuristic algorithms for combining search engine results. The assumption of the algorithms are based on a widely used notion that when different search engines give the same peptide identification, the identification is more likely to be correct [RMAM$^+$04].

A heuristic combiner must choose two outputs. First, the combiner must decide which peptide to assign to each spectrum, choosing among all the peptide IDs for that spectrum produced by the individual search engines. We tested combiners

that selected peptide assignments based on either consensus (voting) or purely based on peptide ID scores. Second, the combiner must assign a score to the peptide ID selected. We tested combiners that used either E-values or eFDR scores, selected in a variety of ways.

Except for MIN-Evalue and MIN-eFDR, heuristic combiners are majority voting based, where peptide IDs for a spectrum are chosen based on the largest number of agreeing search engines. For instance, for a given spectrum if two search engines select peptide x and one search engine selects peptide y, a voting heuristic always assigns peptide x to the spectrum, regardless of the scores assigned by each search engine (even if the peptide ID for y has a much higher confidence score than the two peptide IDs for x).

Note that for our experiments, since we only use results from three search engines there can be only 1-1-1 (three way) or 1-1 (two way) ties in voting. When breaking ties we can simply use the E-value or eFDR score of the single search engine peptide ID in each voting bloc.

Initially, peptide IDs are gathered for all spectra from all search engines. Either the search engine E-value or its eFDR score is used. Estimated FDR scores are calculated for each search engine, per spectrum, based on search results for a decoy (shuffled) database. The differences between voting heuristics are in how ties are broken, and what scores are assigned to the selected peptide ID. The following heuristics were tested:

MIN-Evalue Combiner. Select peptide ID with the minimum (best) search engine E-value, regardless of number of votes. Does not rely on voting or consensus.

MIN-eFDR Combiner. Select peptide ID with the minimum (best) estimated FDR, regardless of number of votes. Does not rely on voting or consensus.

V-Evalue Combiner. Select peptide ID with the largest number of votes. Break ties by selecting peptide ID with minimum (best) search engine E-value. Assign score as minimum E-value among agreeing peptide IDs. Intuitively, the E-value combiner relies on the accuracy of unnormalized scores output by each search engine. Search engines reporting more optimistic scores will have a greater effect on the scores output by the E-value combiner heuristic.

V-MIN eFDR Combiner. Select peptide ID with the largest number of votes. Break ties by selecting peptide ID with minimum (best) eFDR. Assign score as minimum eFDR among agreeing peptide IDs. Intuitively, selecting the minimum eFDR represents an optimistic estimate of statistical significance of the selected peptide ID.

V-MAX eFDR Combiner. Select peptide ID with the largest number of votes. Break ties by selecting peptide ID with minimum (best) eFDR. Where there are multiple agreeing peptide IDs, assign score as the maximum eFDR among the agreeing peptide IDs. Intuitively, selecting the maximum eFDR represents a conservative estimate of statistical significance of the selected peptide ID. Note that V-MIN and

V-MAX will always assign the same peptide ID to a spectrum. The only difference is in the score assigned to the peptide ID.

V-Random Combiner. Select peptide ID with the largest number of votes. Break ties by randomly selecting peptide ID with equal probability. Assign score by randomly selecting scores from agreeing peptide IDs (with equal probability). Intuitively, the random combiner represents a middle-of the road estimate of statistical significance of the selected peptide ID.

Pre-determined Combiners. Select peptide ID with the largest number of votes. Break ties based on a predetermined ordering between search engines. V-TMO (Tandem > Mascot > OMSSA), V-MTO (Mascot > Tandem > OMSSA), and V-OMT (OMSSA > Mascot > Tandem) represent three possible fixed orderings between search engines. Assign score among agreeing peptide IDs using the same ordering. Intuitively, the pre-determined combiner represents a preference for a particular search engine.

Vote-3 Combiner. Similar to the V-MIN eFDR combiner, but only selects peptide IDs if all three search engines agree on the peptide selected. Intuitively, the Vote-3 Combiner is the most conservative consensus-based combiner.

Vote-2 Combiner. Similar to the V-MIN eFDR combiner, but only selects peptide IDs if all three search engines agree on the peptide selected, or if two search engines select the same peptide and the third search engine does not select any peptide.

Intuitively, the Vote-2 Combiner is a conservative consensus-based combiner which allows a single search engine to veto any peptide ID if it chooses a different peptide for a spectrum.

3.3.5 PepArML Framework

3.3.5.1 Supervised Machine Learning Algorithm

The machine learning combiner PepArML (Peptide identification Arbiter by Machine Learning) models the peptide identification problem as a classification problem. PepArML classifies each spectrum's peptide IDs, generated by one or more search engines, as either *true* or *false* with some confidence. For each spectrum, the peptide ID classified as true with highest confidence is the predicted correct peptide ID.

Feature vector. In addition to raw scores and E-values, search engines compute (and report) many additional metrics characterizing peptide IDs (e.g., number of matched ions). Using a machine learning framework makes it possible to use these additional metrics to predict the correctness of peptide IDs with little additional effort. A feature vector is constructed for each peptide identification, per spectrum (peptide-spectrum pair). Figure 3.11 shows the structure of vector. It consists of scores, E-values and additional metrics from each search engine. When search engines assign spectra to different peptides, the score, E-value, and other metrics computed by a search engine may be absent for particular peptide-spectrum pair, in

which case the values are set to sentinels. An additional sentinel feature, per search engine, is set to indicate the presence or absence of the search engine's features for a particular peptide-spectrum pair. Scores and other metrics can be extracted or computed directly from the search engines' outputs, and easily added to the feature vector. At current stage of PepArML design, there is no effort made to eliminate features that may be correlated or have poor predictive performance. Current peptide ID feature vector is shown in Table 3.5.

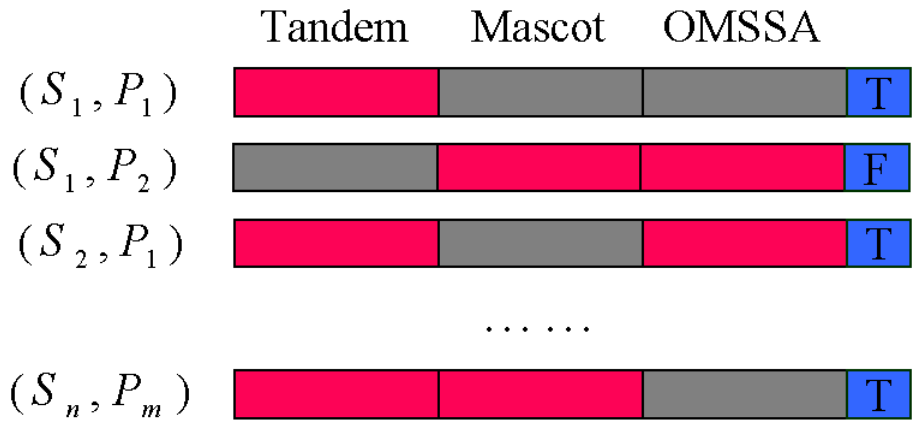

Figure 3.11: Feature Vector

Random Forest training (5-fold) and testing. For PepArML we chose to use the Random Forest algorithm, developed by Leo Breiman, a statistician at the University of California Berkeley. It is a meta machine learning classifier which builds multiple decision trees. Each tree is built with random training samples and randomly selected attributes set at each tree node. The final decision is made by the result of majority vote. We chose Random Forest because it doesn't impose any statistical models on the attribute distribution, and thus should yield better oppor-

Search Engine	Sequence #	**Description**
—	1	Peptide length
Tandem	2	Hyperscore
	3	Precursor mass delta
	4	# of matched y-ions
	5	# of matched b-ions
	6	# of missed cleavage
	7	Sum of matched intensity
	8	E-value
	9	Sentinel
Mascot	10	Score
	11	Precursor mass delta
	12	# of matched ions
	13	# of matched peaks
	14	# of missed cleavages
	15	E-value
	16	Sentinel
OMSSA	17	p-value
	18	# of matched ions
	19	E-value
	20	Sentinel

Table 3.5: Peptide ID feature vector, with features from Tandem, Mascot, and OMSSA.

tunities for actual data properties to decide the peptide identification results.

The classifiers were trained with all possible combinations of search engines. Classifiers using only the features of Mascot, OMSSA, and Tandem, are denoted C-M, C-O, and C-T, respectively. Classifiers C-MO, C-TM, and C-TO use features from Mascot and OMSSA; Tandem and Mascot; and Tandem and OMSSA, respectively. Classifier C-TMO uses features from all three search engines. Each classifier is trained and tested using 5-fold cross validation. Since the number of false peptide IDs is larger than the number of true peptide IDs, the false peptide IDs were down-sampled before training so that each training data set has equal number of true and false examples.

3.3.5.2 Unsupervised Learning

For tandem mass spectra data sets derived from synthetic protein mixes, training peptide IDs can be classified as true or false based on the identity of proteins known to be in a sample. In practice, however, the protein content of biological samples is not known in advance. Peptide ID training sets thus cannot be constructed based on ground truth. One of the major novel features of PepArML is an unsupervised training procedure developed for this case.

The unsupervised training procedure relies on two key observations. First, many proteins in a sample can typically be confidently identified by database search engines, even if not all the peptides or spectra from these proteins can be confidently identified. For instance, in experiments biologists frequently consider a protein to

be positively identified if two or mass spectra are assigned with high confidence to peptides from that protein.

Once a protein is positively identified (based on some quality criteria), PepArML can then also label all other peptide IDs assigned to the protein as correct peptide IDs in the training data. This property is where peptide identification differs from generic machine learning classification problems. Since a single protein sequence is longer than and can thus contain multiple short peptide sequences, assuming a protein to be correct based on two or more peptide IDs allows us to potentially assign true labels to many other peptide IDs for the protein, despite their lower confidence.

The second observation on which the PepArML unsupervised training procedure is based is that for peptide identification, machine learning models can be successfully trained even if the training labels are not completely correct. Classification results from interim machine learning models may be used to re-label training data and iteratively build more accurate models. PepArML takes advantage of this observation to begin training models initially based on one or more high-confidence putative true proteins, then iteratively improve the accuracy of each model by using its classification results to label training data for newer models.

3.3.5.3 Unsupervised Learning Algorithm

The unsupervised peptide identification is carried out in an iterative procedure. A set of putative true positive proteins are selected based on input search engines' results, and are used to label peptide IDs. Machine learning is then applied and

a peptide ID classifier is trained. The trained peptide ID classifier is then used to select new putative true positive proteins. The procedure is then iterated until convergence. The algorithmic steps are:

Select putative true positive proteins by consensus. First, consensus peptides that all search engines agree on are selected as initial *putative* true peptides. Proteins that contain at least two such peptides and have a significant fraction of their sequence (e.g., 10%) covered by these peptides are selected as initial putative true positive protein set.

Iteratively refine the putative true positive proteins. Given putative true positive proteins, all peptide IDs associated with these proteins are labeled true, with all other peptide IDs labeled false. A classifier is trained based on the labeled data with 5-fold cross-validation. Peptide IDs predicted as true, below some estimated FDR value (e.g., 0.2), are selected. Proteins that contain at least two such peptides and have certain sequence (e.g., 10%) covered by predicted true peptides are selected to form a new putative true positive protein set.

Termination. The iterative procedure continues until the content of putative true positive protein set is stable.

We emphasize that the set of putative true positive proteins need not to be completely correct in order for this iterative training procedure to be successful. In fact, we will show later in Section 3.3.8.8 that the unsupervised PepArML iterative training method can robustly handle omission of true positive proteins, even to the

extent of starting training with only peptide IDs from a single true protein labeled as true.

3.3.6 Implementation

PepArML is implemented using the software package *Weka* [WF05]. (Waikato Environment for Knowledge Analysis). Weka was developed by a machine learning research group at the University of Waikato at New Zealand for data mining. The whole package is written in Java, and contains tools for data pre-processing, classification, regression analysis, clustering, association rules, and visualization. The code was downloaded from `http://www.cs.waikato.ac.nz/ml/weka/`. We selected Weka because of its popularity in the machine learning community.

Weka contains implementations of a number of machine learning classification algorithms. We tested several, including Logistic Regression, support vector machines (SVM), Random Forest, AdaBoost, and naive Bayes. We chose to use the Random Forest classifier for PepArML since it consistently outperformed the others.

3.3.7 Experimental Framework

3.3.7.1 Mass Spectra Data Sets

Three LC-MS/MS data sets were selected for evaluating the proposed approach. They were generated with known protein mixtures by three different instruments. The instruments use a variety of ionization, mass measurement, and fragmentation technologies. The data sets are labeled as C8, S17, and AURUM.

C8 A mixture of 20ng each of eight protein standards (Table 3.6 lists the 8 proteins in the sample plus Trypsin) was prepared. Urea and DTT were added with final concentrations of 8 M and 1 mg/mL, respectively, and incubated at 37℃ for 2 hr under nitrogen. Iodoacetamide was added to a concentration of 2 mg/mL and kept at room temperature for 1 hr in the dark. Trypsin was added at a 1:20 (w/w) enzyme to substrate ratio and incubated overnight at 37℃. The protein digest was desalted using a reversed-phase trap column (Michrom Bioresources, Auburn, CA) and lyophilized to dryness using a SpeedVac (ThermoSavant, San Jose, CA), and then stored at -80℃.

For LC-MS the peptide mixture was injected onto a trap column (3 cm x 200 micron i.d. x 365 micron o.d.) packed with 5 micron porous C18 reversed-phase particles. The peptide fraction was subsequently analyzed by nano-RPLC equipped with an Ultimate dual-quaternary pump (Dionex, Sunnyvale, CA) connected to a fused-silica capillary (50 micron i.d. x 365 micron o.d.). This 15-cm long capillary was packed with 3-micron Zorbax Stable Bond (Agilent, Palo Alto, CA) C18 particles.

Nano-RPLC separation was performed at a flow rate of 200 nL/min using a 5-45% linear acetonitrile gradient over 100 min with the remaining 20 min for column regeneration and equilibration. The peptide eluants were monitored using a linear ion-trap mass spectrometer (LTQ, ThermoFinnigan, San Jose, CA) operated in a data dependent mode. Full scans were collected from 400–1400 m/z and 5 data dependent MS/MS scans were collected with dynamic

exclusion set to 30 sec.

C8 contains a total of 3812 MS/MS spectra, including 504 (13.2%) true positive spectra assignable to peptides from the expected proteins.

S17 1244 MS/MS spectra from the Sashimi project data repository (`http://sashimi.sourceforge.net`) data set `17mix_test2`, representing a tryptic digest of standard proteins (Table 3.7 lists proteins in the sample) analyzed with a Micromass Q-TOF Ultima. The S17 data set contains 247 (19.9%) true positive spectra assignable to peptides from the expected proteins.

AURUM 10082 MS/MS spectra from the Aurum 1.0 data set [FVK+07]. Spectra were generated using a MALDI TOF-TOF instrument, (Applied Biosystems 4700) from 246 commercially sourced human proteins synthetically expressed in E. coli, checked for purity, digested with trypsin, and spotted, one protein per MALDI spot, for analysis. All spectra from spots with AURUM peptide identification annotations were used. The AURUM data set contains 4061 (40.3%) true positive spectra assignable to peptides from the AURUM annotation proteins.

	Name	Accession	Description	Organism
1.	ADH1_YEAST	P00330	Alcohol dehydrogenase 1	S. cerevisiae
2.	PYGM_RABIT	P00489	Glycogen phosphorylase	O. cuniculus
3.	CAH2_BOVIN	P00921	Carbonic anhydrase II	B. taurus
4.	ALBU_BOVIN	P02769	Serum albumin	B. taurus
5.	SODC_BOVIN	P00442	Superoxide dismutase	B. taurus
6.	CYC_HORSE	P00004	Cytochrome c	E. caballus
7.	MYG_HORSE	P68082	Myoglobin	E. caballus
8.	RNAS1_BOVIN	P61823	Ribonuclease pancreatic	B. taurus
9.	ADH2_YEAST	P00331	Alcohol dehydrogenase 2	S. cerevisiae
10.	TRYP_PIG	P00761	Trypsin	S. scrofa

Table 3.6: Proteins in Calibrant protein mixture (C8).

	Name	Accession	Description	Organism
1.	AMY_BACLI	P06278	Alpha-amylase	B. licheniformis
2.	BGAL_ECOLI	P00722	Beta-galactosidase	E. coli
3.	PPB_ECOLI	P00634	Alkaline phosphatase	E. coli
4.	OVAL_CHICK	P01012	Ovalbumin	G. gallus
5.	G3P_RABIT	P46406	Glyceraldehyde-3-phosphate dehydrogenase	O. cuniculus
6.	MLE1_RABIT	P02602	Myosin	O. cuniculus
7.	PYGM_RABIT	P00489	Glycogen phosphorylase	O. cuniculus
8.	ALBU_BOVIN	P02769	Serum albumin	B. taurus
9.	MYSS_RABIT	P02562	Myosin heavy chain	O. cuniculus
10.	CAH2_BOVIN	P00921	Carbonic anhydrase	B. taurus
11.	CASB_BOVIN	P02666	Beta-casein	B. taurus
12.	CATA_BOVIN	P00432	Catalase	B. taurus
13.	LACB_BOVIN	P02754	Beta-lactoglobulin	B. taurus
14.	LALBA_BOVIN	P00711	Alpha-lactalbumin	B. taurus
15.	TRFE_BOVIN	Q29443	Serotransferrin	B. taurus
16.	ACTA_HUMAN	P62736	Actin	H. sapiens
17.	ALBU_HUMAN	P02768	Serum albumin	H. sapiens
18.	MYG_HORSE	P68082	Myoglobin	E. caballus
19.	PHS2_RABIT		Glycogen Phosphorylase	O. cuniculus
20.	MLRS_RABIT	P02608	Myosin regulatory light chain 2	O. cuniculus
21.	MLRT_RABIT	P24732	Myosin regulatory light chain 2	O. cuniculus
22.	MYH2_RABIT		Myosin	O. cuniculus
23.	MYH1_RABIT		Myosin	O. cuniculus
24.	MYH13_RABIT		Myosin	O. cuniculus
25.	MYH4_RABIT	Q28641	Myosin-4	O. cuniculus
26.	MYH8_RABIT		Myosin	O. cuniculus

Table 3.7: Proteins in Sashimi protein mixture (S17).

3.3.7.2 Tandem Mass Spectra Search Engines and Protein Sequence Database

Three sequence database search engines were used to identify the tandem mass spectra: X!Tandem [CB04], Release 2007.07.01; MASCOT [PPCC99], version 2.1.03; and OMSSA [GMK+04], version 2.1.0. Search parameters are summarized in Table 3.8. For X!Tandem, refinement mode is turned off to make sure X!Tandem doesn't take advantage of the refined search procedure.

Two protein sequence databases were used: UniProtKB/Swiss-Prot (version 53.0) and Human International Protein Index (IPI-Human) (version 3.32). Data sets C8 and S17 were searched against Swiss-Prot, while data set AURUM was searched against IPI-Human. For each sequence database, a decoy database was also created. It consists of randomly shuffled sequences. Search results from the decoy database were used for estimating false discovery rates (FDR), using the method described by Elias *et al.* [EHFG05a].

When searching tandem mass spectra data sets against protein sequence databases, each search engine generates zero, one, or more peptide IDs for each spectrum. Also because each search engine has its own algorithm for selecting candidate peptides, filtering noisy spectra, assigning peptide ID score and assessing the quality and statistical significance of peptide-spectrum match, the peptide IDs are different among search engines.

For the proposed approach, only the top ranked peptide ID is extracted from each search engine for each spectrum. Peptide IDs corresponding to an expected

protein, which belongs to the synthetic protein mix or the selected putative true protein set, are considered correct. The vector corresponding to this peptide is assigned as "true", regardless of E-value, score or number of agreeing search engines. All other peptide IDs are considered incorrect and the corresponding vectors are assigned as "false".

Search Parameter	**C8**	**S17**	**AURUM**
Data Base	Swiss-Prot	Swiss-Prot	IPI_Human
Mass Error Unit	Dalton	Dalton	Dalton
Fragment Mass Tolerance	0.6	0.2	0.4
Parent Mass Tolerance	2.0	2.0	2.0
Maximum Missed Cleavages	1	1	1
Enzyme	Trypsin	Trypsin	Trypsin
Fixed Modification	Carbamidomethyl(C)	Carbamidomethyl(C)	Carbamidomethyl(C)
Variable Modification	Oxidation(M)	Oxidation(M)	Oxidation(M)

Table 3.8: Search Engine Parameters for Spectra Data Sets

3.3.8 Experimental Results

3.3.8.1 Search Engine E-value vs. Estimated False Discovery Rate

We first briefly examine the peptide identification scoring technique for each search engine. In Figure 3.12, the left column compares each search engine's estimated statistical significance (E-values) and false discovery rate (FDR). The results are collected based on three spectra data sets searched with three search engines. Since E-value for certain score S is defined as the estimated number of false peptide identifications with score equal or better than S, and FDR is defined as the percentage of false positives among the declared significant results, E-value and FDR represent similar concepts when measuring the sensitivity of peptide identification results.

Ideally, there should be linear correspondence between E-values calculated by each search engine and experimentally measured FDR. However, we see in Figure 3.12 that the actual result are quite different. E-values for both Tandem and Mascot tend to increase quickly for low FDR values, then level out and grow much more slowly. OMMSA E-values are linear but fairly flat. In addition, we see that E-values do not correspond to actual measured FDR. For instance, an E-value of 0.5 implies that half of all predictions are correct. However, the graph for each search engine E-value=0.5 does not yield the appropriate FDR for each data set.

Comparing E-values produced by different search engines is also difficult. Because the statistical models for estimating E-values are different in each search engine, the same E-value given by different search engines may correspond to differ-

ent FDR. In practice results show E-values of search engines were frequently overly optimistic or pessimistic. E-values thus do not appear to be sufficiently precise to estimate the number of false hits by each search engine. Even worse, not only is the correlation between E-values and FDR poor, the relative accuracy for each search engine changes for different data sets. We can conclude that E-values are unlikely to be a good metric for comparing search engine performance or combining multiple search engine results.

In comparison, we found that estimated FDR (eFDR) values were better correlated to true FDR values. The right column of Figure 3.12 presents the true FDR values for corresponding eFDR values for all three data sets. The correlation between eFDR and FDR is much better than between E-values and FDR. We see that eFDR values computed from decoy databases are somewhat pessimistic (under-predicting true FDR) but generally linearly correlated, and more consistent than E-values across all three data sets. Estimating FDR based on experimental decoy (reverse/shuffle) database search thus appears to be a better approach for predicting the statistical significance of peptide IDs.

3.3.8.2 Heuristic Combiner Comparisons

Next, we compare the sensitivity of the different heuristics for combining search engine results. Figures 3.13, 3.14 and 3.15 show the ROCs of different heuristics for each mass spectra data set. Table 3.9 lists sensitivity at 10% and 20% FPR for each heuristic combiner.

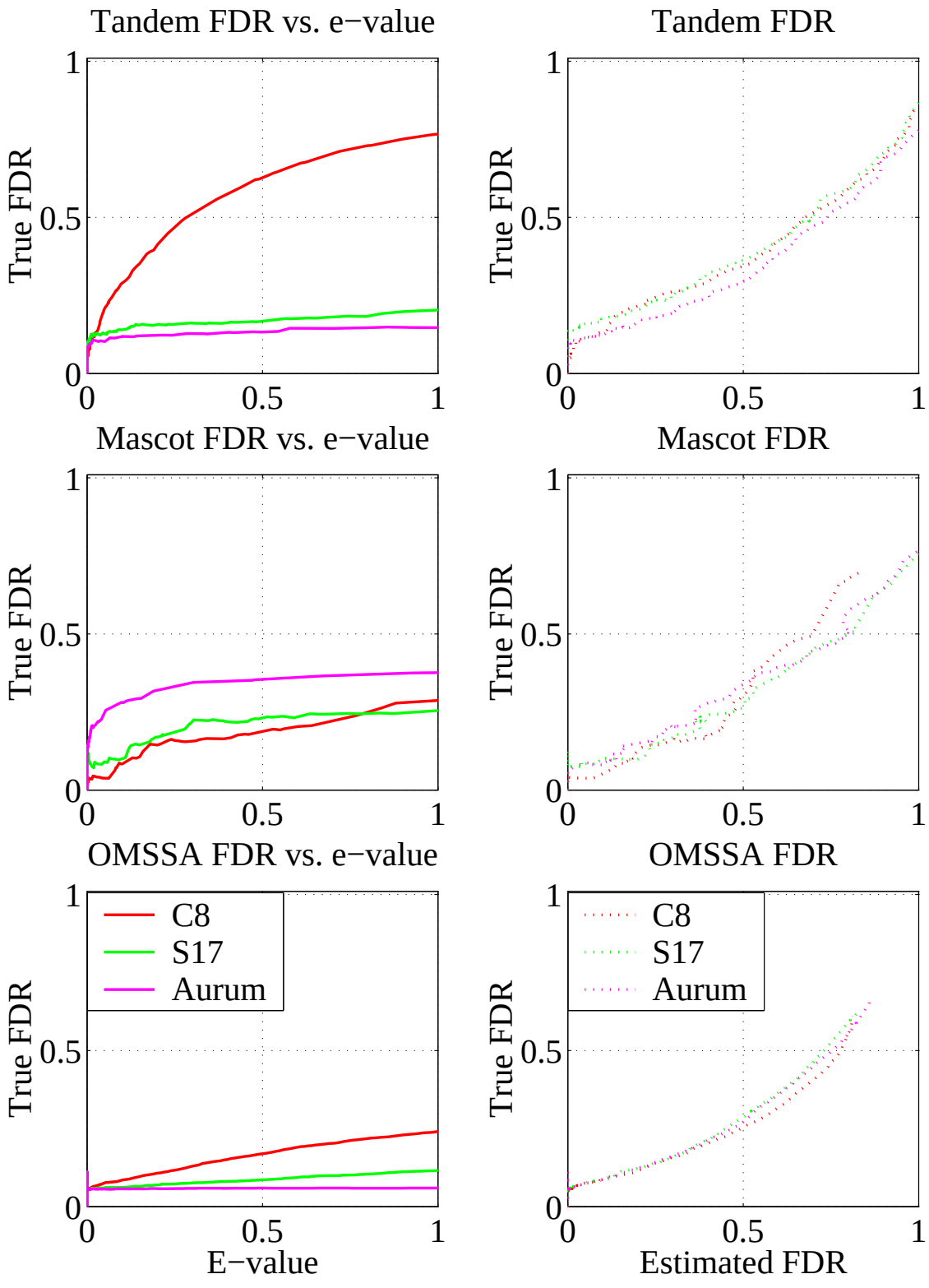

Figure 3.12: Correlation between FDR, E-value and estimated FDR

The main observation is that both the absolute and relative performance of each heuristic combiner is data set dependent. For C8 V-MIN and MIN-eFDR tend to be the most sensitive, and Vote-3 and V-MAX the least. For S17 Vote-3 and Vote-2 are the most sensitive, while V-MAX is significantly worse than all other combining heuristics (except for very small FPR values, where it is the best). For AURUM V-MIN and MIN-eFDR are again the most sensitive. V-MAX and V-OMT are the least sensitive, but only by a small margin.

Overall, we see that the V-MAX combiner tends to be less accurate than the other heuristics. Apparently always conservatively selecting higher (worse) eFDR values for its output appears to reduces the heuristic's sensitivity. Vote-3 appears to be too conservative for C8, but not for the other data sets.

The performance of the voting heuristic V-MIN and the non-voting heuristic MIN-eFDR are very similar. Both heuristics choose the minimum (best) eFDR value, but V-MIN only does so among the peptide IDs with the greatest agreement. For our datasets the similarity in the output of the two heuristics indicate the best eFDR values usually occurs only in the largest group of agreeing peptide IDs. In other words, agreement and high eFDR scores are very well correlated in our data sets.

Similar to V-MIN and Min-eFDR, the sensitivity of the voting heuristic V-Evalue and the non-voting heuristic MIN-Evalue are also very similar. Both heuristics choose the minimum (best) E-value, but V-Evalue only does so among the peptide IDs with the greatest agreement.

The power of consensus is demonstrated by the sensitivity of the Vote-3 and

Vote-2 heuristics for the S17 data set, where for FPR values between 0.02 and 0.1 either Vote-3 or Vote-2 is significantly more sensitive than the other heuristics. However, Vote-3 and Vote-2 do not perform as well for the C8 and AURUM data sets.

In most cases, results seem to indicate that except for V-MAX, the performance of the eFDR based heuristic combiners (V-MIN, V-Random, V-TMO, V-MTO, and V-OMT) are about the same, with V-MIN slightly more sensitive overall. This seems to indicate that search engine eFDR scores are fairly similar when the same peptide ID is selected.

Finally, comparing the use of E-values (V-Evalue and MIN-evalue) versus eFDR scores (V-MIN, MIN-eFDR, etc.), we see that heuristics using E-values perform well for AURUM (tied with eFDR) but are slightly less sensitive for C8 and S17.

Since the V-MIN combiner usually achieves the best sensitivity, we will refer to it as *Voting* and use it as the representative voting heuristic for comparisons with machine learning methods in the remainder of the chapter.

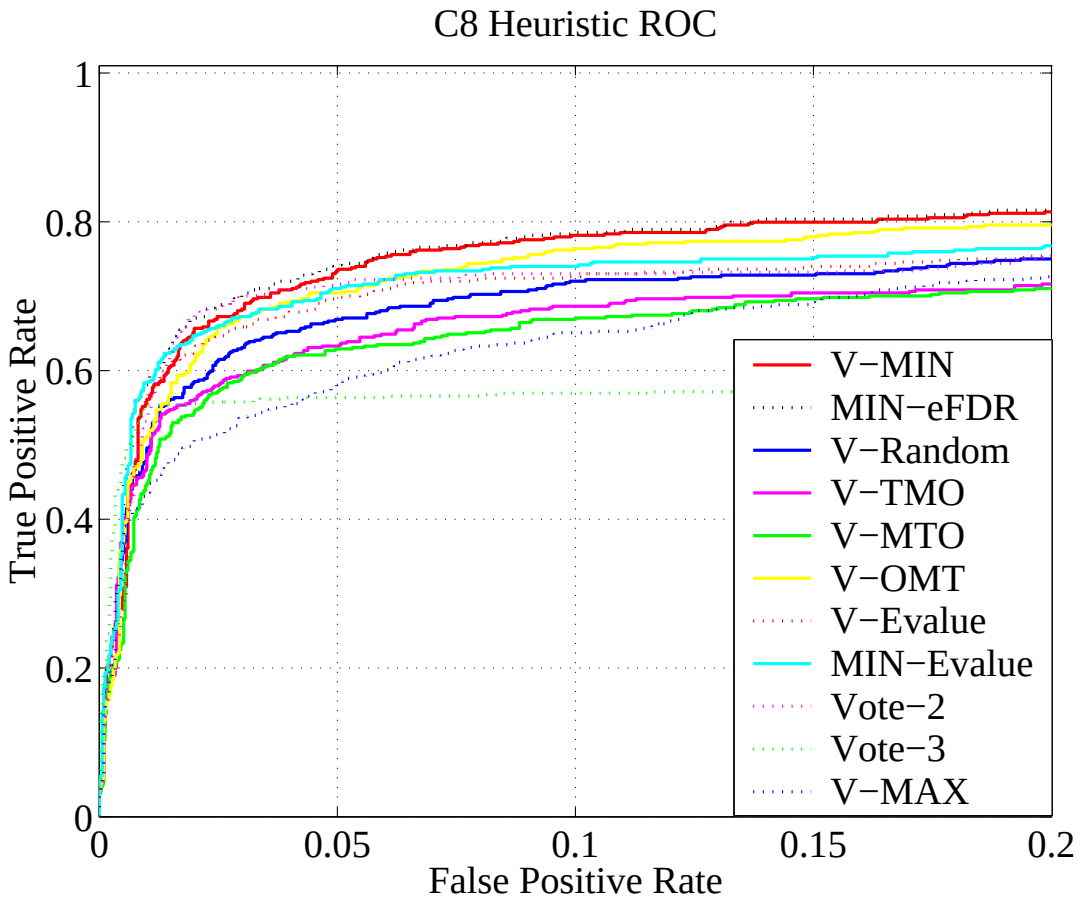

Figure 3.13: Heuristics for C8 Spectra Set

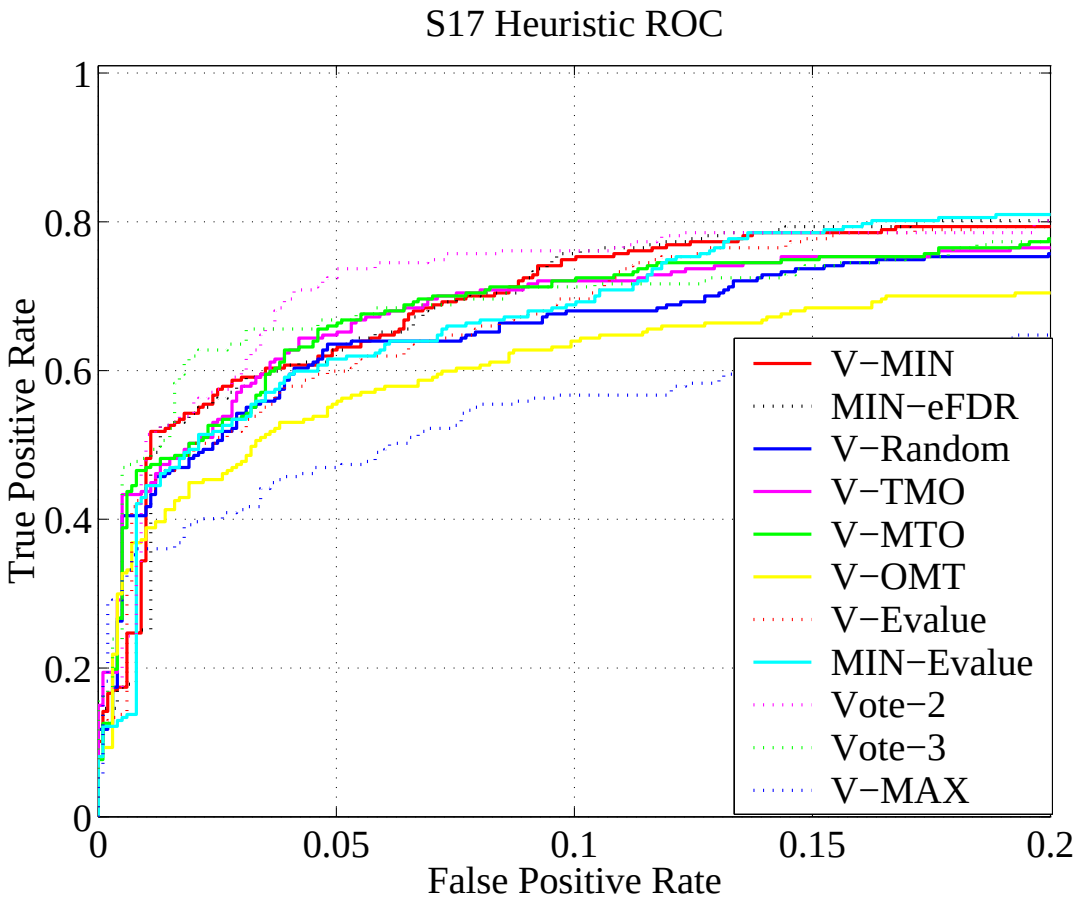

Figure 3.14: Heuristics for S17 Spectra Set

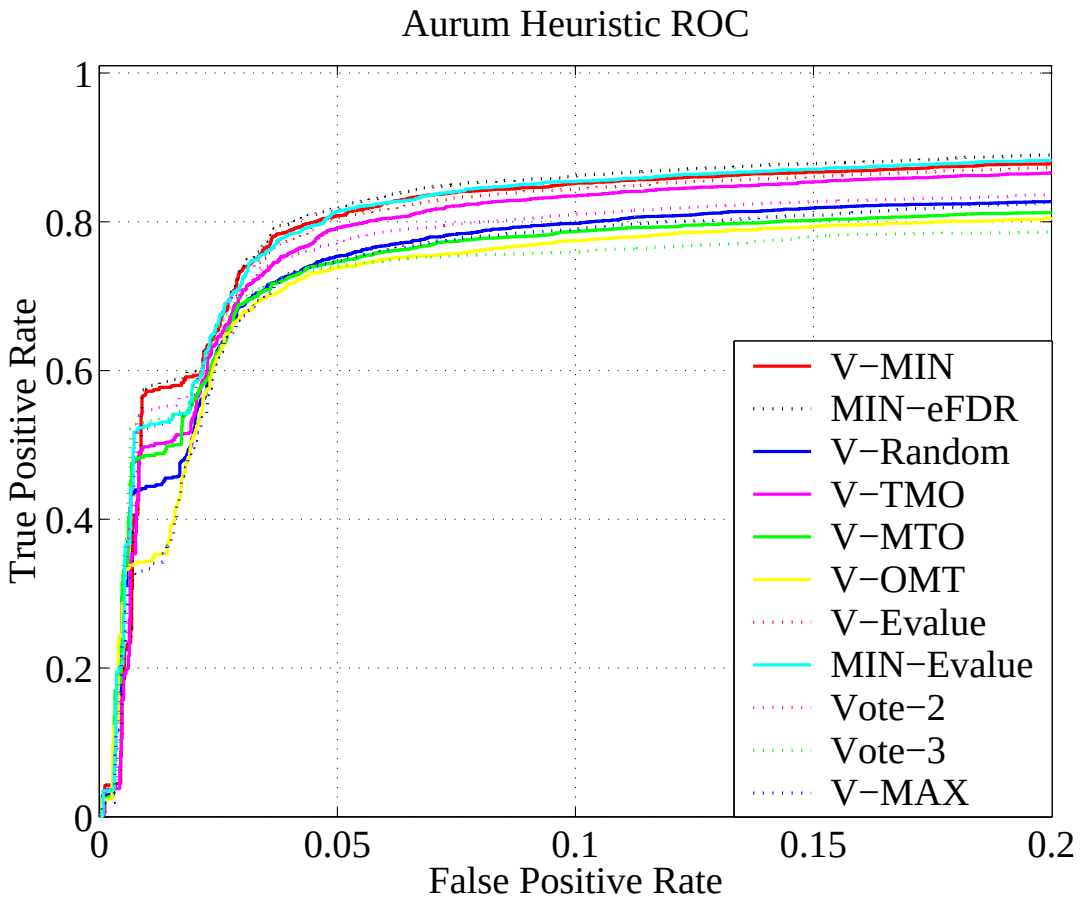

Figure 3.15: Heuristics for AURUM Spectra Set

	C8		S17		AURUM	
Classifiers	**10% FPR**	**20% FPR**	**10% FPR**	**20% FPR**	**10% FPR**	**20% FPR**
MIN-Evalue	0.74	0.77	0.69	**0.81**	0.85	0.88
MIN-eFDR	**0.78**	**0.82**	**0.76**	0.80	**0.86**	**0.89**
V-Evalue	0.73	0.75	0.70	0.79	0.84	0.87
V-MIN	**0.78**	0.81	0.75	0.79	0.85	0.88
V-MAX	0.65	0.73	0.57	0.65	0.79	0.83
V-Random	0.72	0.75	0.68	0.76	0.80	0.83
V-TMO	0.69	0.72	0.72	0.77	0.83	0.87
V-MTO	0.67	0.71	0.72	0.78	0.79	0.81
V-OMT	0.76	0.80	0.64	0.70	0.77	0.80
Vote-2	0.73	0.75	**0.76**	0.80	0.81	0.84
Vote-3	0.57	0.61	0.71	0.77	0.76	0.79

Table 3.9: Sensitivity vs. False Positive Rate (FPR) for Heuristic Combiners. Best sensitivity for each FPR & data set in bold.

3.3.8.3 Voting Heuristics vs. Search Engine Comparisons

We now compare the sensitivity of some voting heuristics (V-MIN, Vote-2, Vote-3) with that of the individual search engines (Tandem, Mascot, OMSSA). Figures 3.16, 3.17, and 3.18 show the resulting ROC curves.

We see that even relatively simple heuristic combiners can improve sensitivity. except for very low FPR values (< 0.01), heuristic combiners are more sensitive than any individual search engine. At higher FPR values, only the Vote-3 heuristic combiner is less sensitive than individual search engines. This result is in line with earlier observations that individual search engines miss many correct peptide IDs found by other search engines. In fact, no single search engine is consistently more sensitive than the others across all three data sets. OMSSA is most sensitive for C8, Mascot for S17, and Tandem for AURUM.

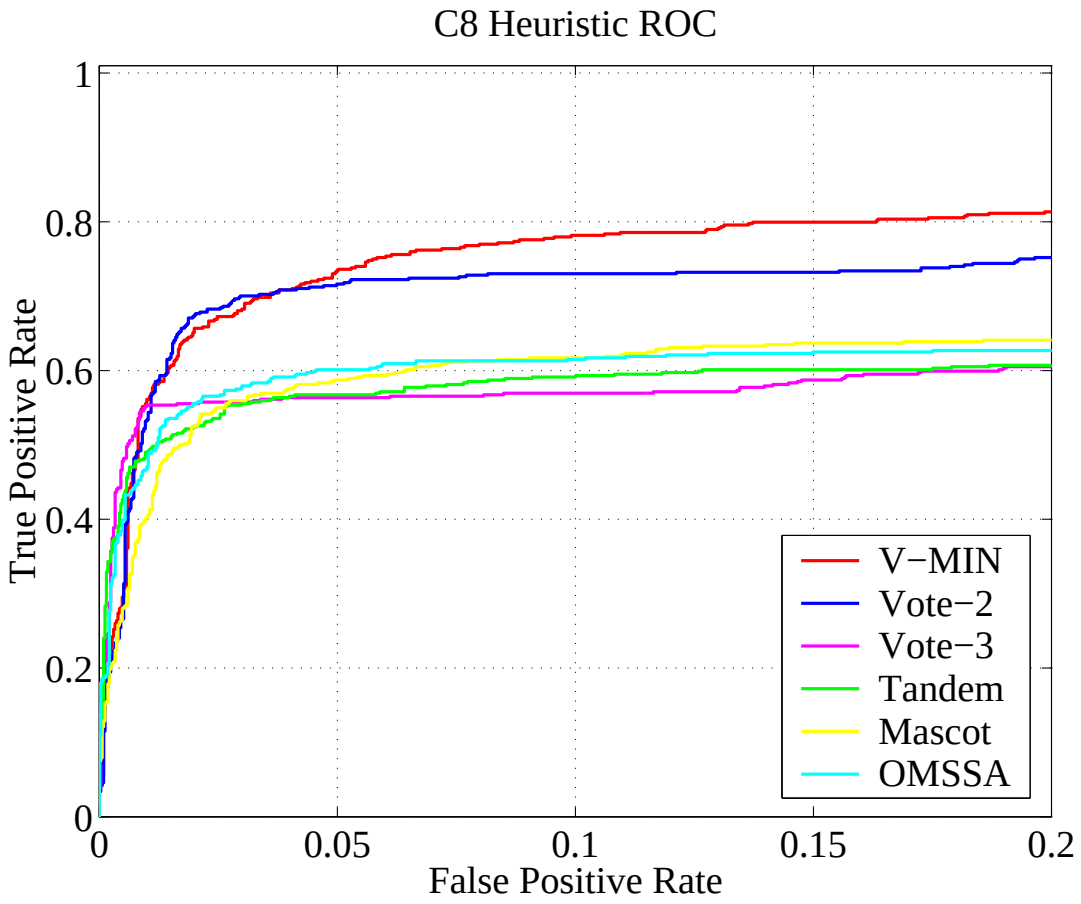

Figure 3.16: Heuristics vs. Single Search Engine Comparison for C8 Spectra Set

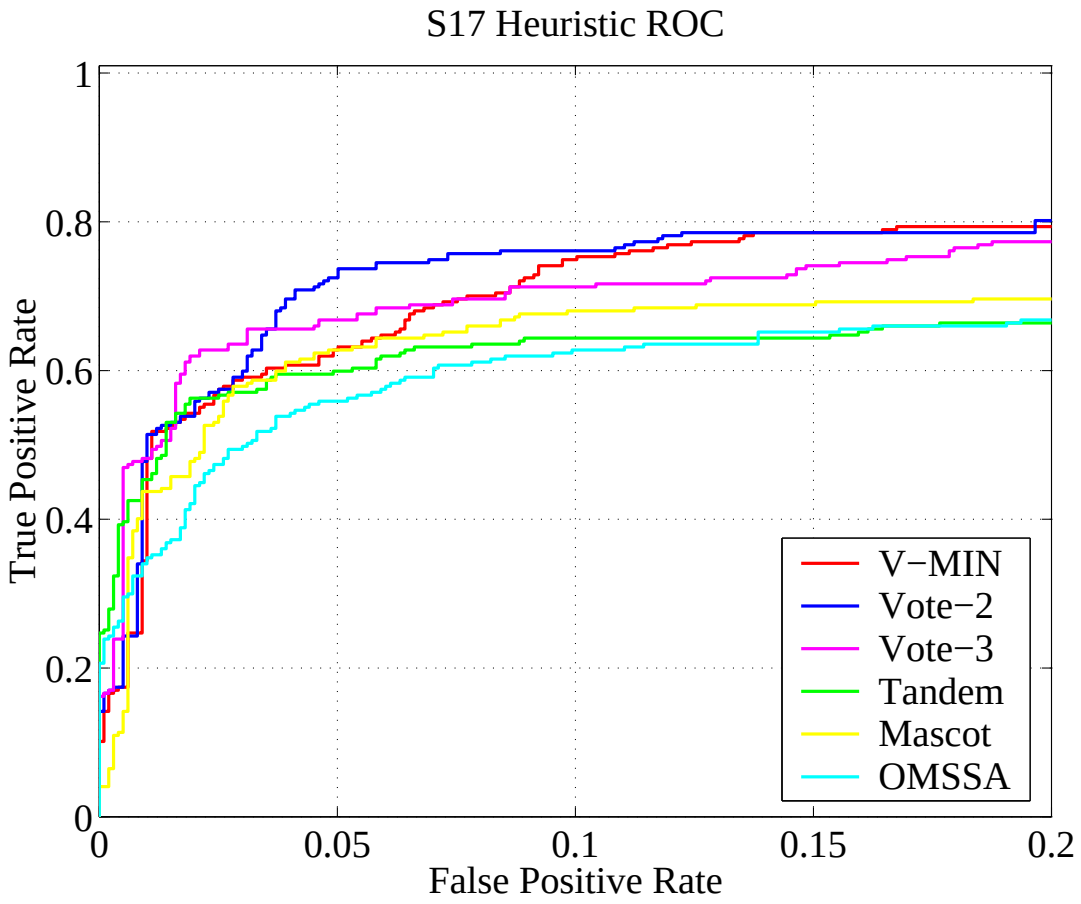

Figure 3.17: Heuristics vs. Single Search Engine Comparison for S17 Spectra Set

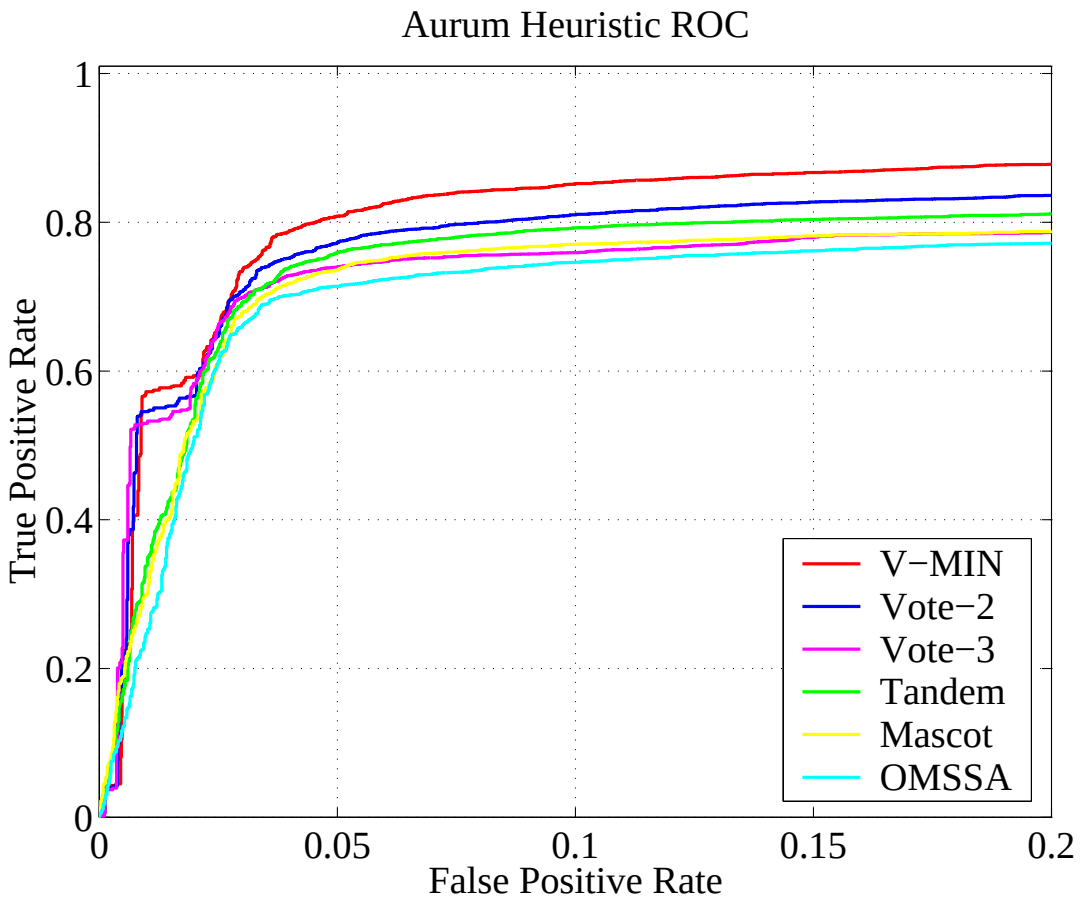

Figure 3.18: Heuristics vs. Single Search Engine Comparison for AURUM Spectra Set

3.3.8.4 Supervised Machine Learning Performance

Having seen that heuristic combiners work quite well, we now explore whether machine learning is really needed to combine search engine results. We present our experimental evaluation in three sets of graphs: ROC curves, sensitivity vs. FDR plots, and sensitivity vs. estimated FDR (eFDR) plots. Results for each data set are placed in six separate graphs to make comparisons easier (with data for the voting heuristic and overall combined classifier C-TMO replicated on all graphs).

ROC Curves

We begin with classic ROC curves that compare sensitivity and selectivity tradeoffs for each algorithm. Classifier and search engine ROC curves for each data set are shown in Figures 3.19 (C8), 3.20 (S17), and 3.21 (AURUM), with six sets of graphs for each data set. In each graph, the y-axis represents true positive rate (sensitivity), and the x-axis represents false positive rate (1-specificity). The three graphs on the left of each figure present ROC curves for classifiers C-T, C-M, and C-O (displayed as solid lines) and for search engines Tandem, Mascot, and OMSSA (displayed as dotted lines). The three graphs on the right of each figure present ROC curves for classifiers C-TM, C-MO, and C-TO (displayed as solid lines). ROC curves for C-TMO (displayed as dash-dotted lines) and Voting (V-MIN eFDR) (displayed as dashed lines) are included in all graphs for comparison. Note that the ROC curves only display smaller false positive rates from 0 to 0.2, which are the most interesting to biologists.

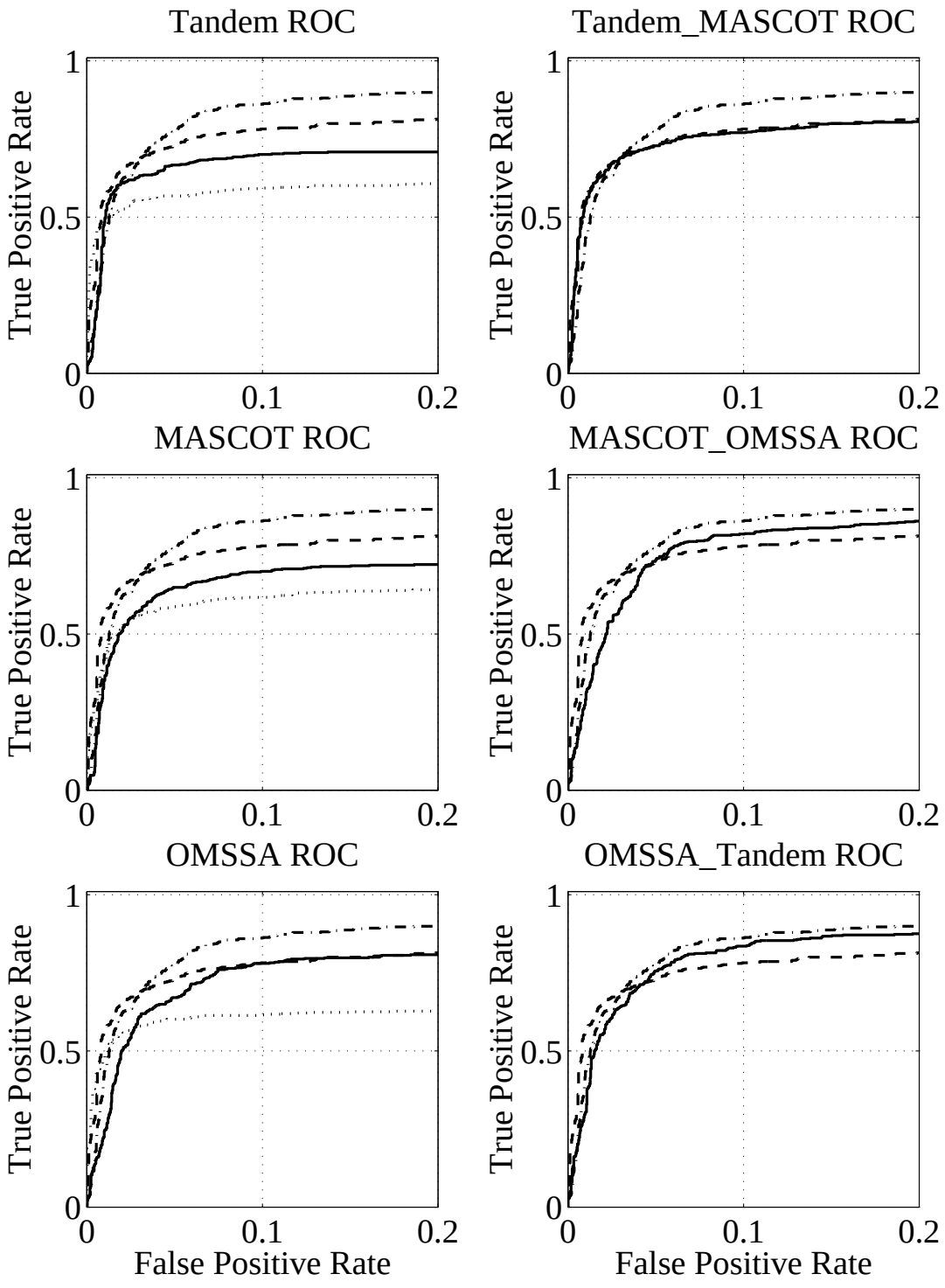

Figure 3.19: ROC curves for C8. Classifiers (solid line) and search engines (dotted line) are presented in each graph. C-TMO classifier (dash-dotted line) and Voting combiner (dashed line) are included in all graphs.

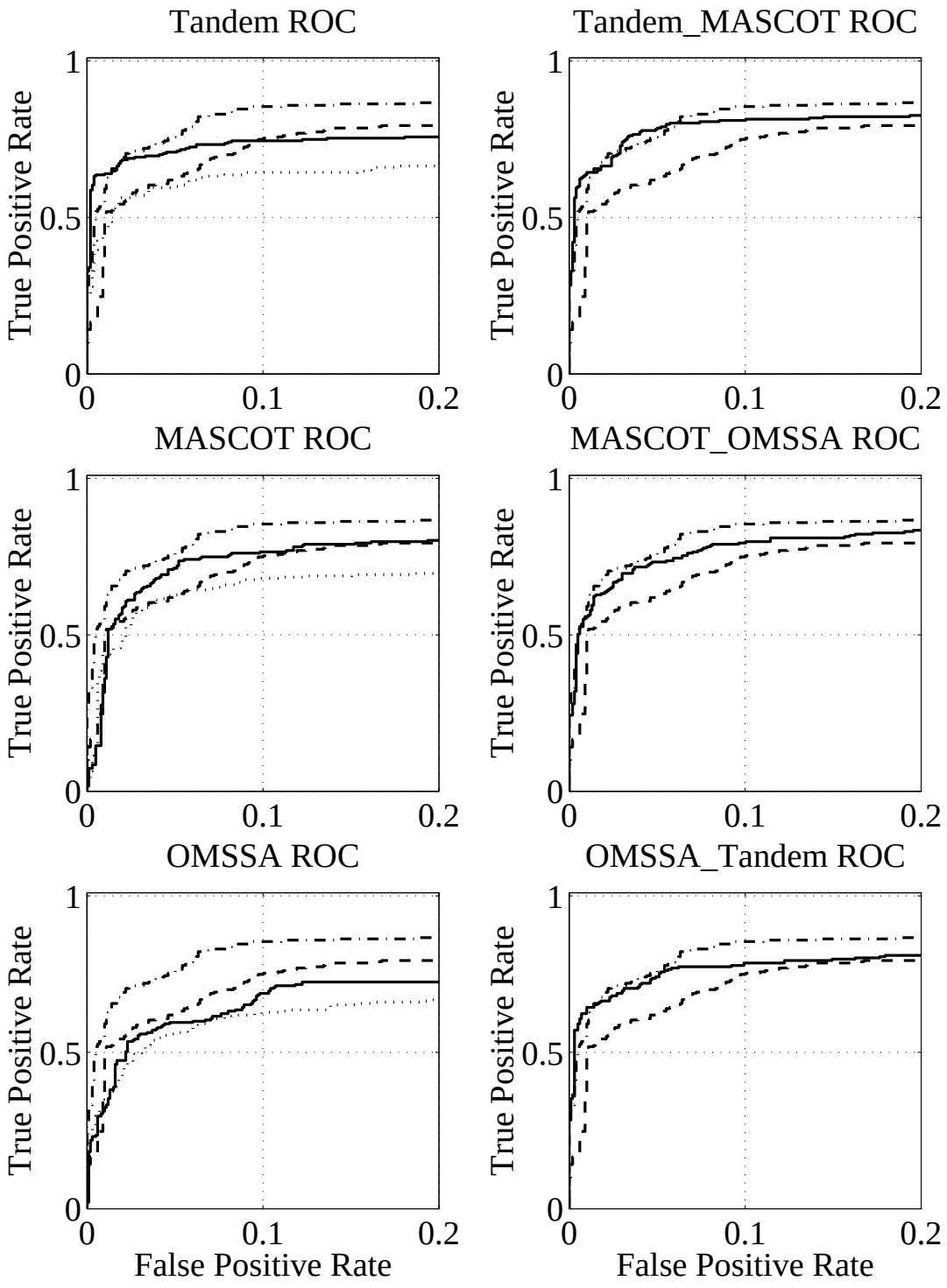

Figure 3.20: ROC curves for S17. Classifiers (solid line) and search engines (dotted line) are presented in each graph. C-TMO classifier (dash-dotted line) and Voting combiner (dashed line) are included in all graphs.

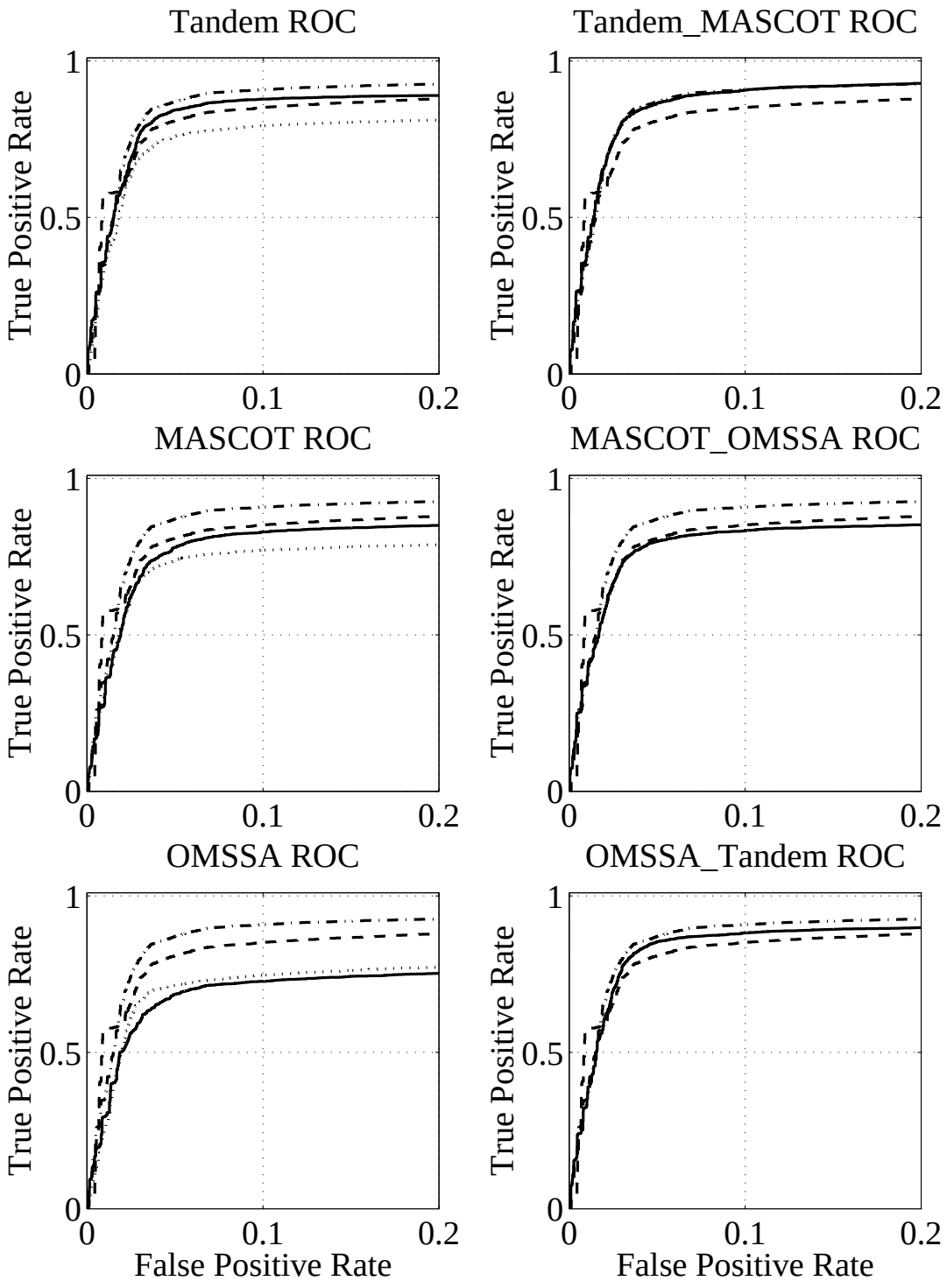

Figure 3.21: ROC curves for AURUM. Classifiers (solid line) and search engines (dotted line) are presented in each graph. C-TMO classifier (dash-dotted line) and Voting combiner (dashed line) are included in all graphs.

	C8		S17		AURUM	
Classifiers	**5% FPR**	**10% FPR**	**5% FPR**	**10% FPR**	**5% FPR**	**10% FPR**
C-TMO	**0.79**	**0.86**	0.76	**0.85**	**0.87**	**0.91**
C-TM	0.73	0.77	**0.79**	0.81	0.86	**0.91**
C-MO	0.74	0.82	0.73	0.80	0.80	0.83
C-TO	0.76	0.84	0.75	0.79	0.86	0.88
C-T	0.67	0.70	0.71	0.74	0.84	0.88
C-M	0.65	0.70	0.71	0.77	0.78	0.83
C-O	0.68	0.78	0.60	0.69	0.69	0.73
Voting	0.73	0.78	0.63	0.75	0.81	0.85

Table 3.10: Sensitivity vs. False Positive Rate (FPR) for Classifiers. Best sensitivity for each FPR & data set in bold.

Table 3.10 presents the sensitivity at 5% and 10% FPR for each classifier and Voting heuristic combiner. These values correspond to the height of each curve at the 0.05 and 0.1 points on the x-axis of each of the ROC graphs.

The main observation is that C-TMO (dash-dotted line), the classifier built using all three search engines, generally yields the best sensitivity overall. Minor exceptions exist for very small values of FPR, particularly when compared to C-T for S17. C-TMO outperforms not just single search engine scores and classifiers, but also the V-MIN heuristic combiner. Differences are small for very small FPR values, but by 10% FPR the increase in sensitivity is significant. Combining results using machine learning thus appears more successful than heuristic combiners.

As previous researchers have discovered, results also show classifiers using only features from a single search engine (solid lines on left) are generally more sensitivity than the original search engines (dotted lines on left). Gains are consistent for C-T and C-M; however, C-O (the classifier built using features from OMSSA) is less sensitive than OMSSA for AURUM and for low FPR values for C8, possibly due to the small number of features output by OMSSA.

Finally, results demonstrate that classifiers built using features from two search engines (C-TM, C-MO, C-TO, solid lines on right) are almost as sensitive as C-TMO. Overall, sensitivity results vary somewhat depending on the data set tested, but general trends remain mostly consistent.

Similar observations may be drawn from the area under ROC (AUROC) plots in Figure 3.22. Note that our ROC graphs display the section of ROC of greater interest to biologist (FPR values between 0 and 0.2), but AUROC is computed as

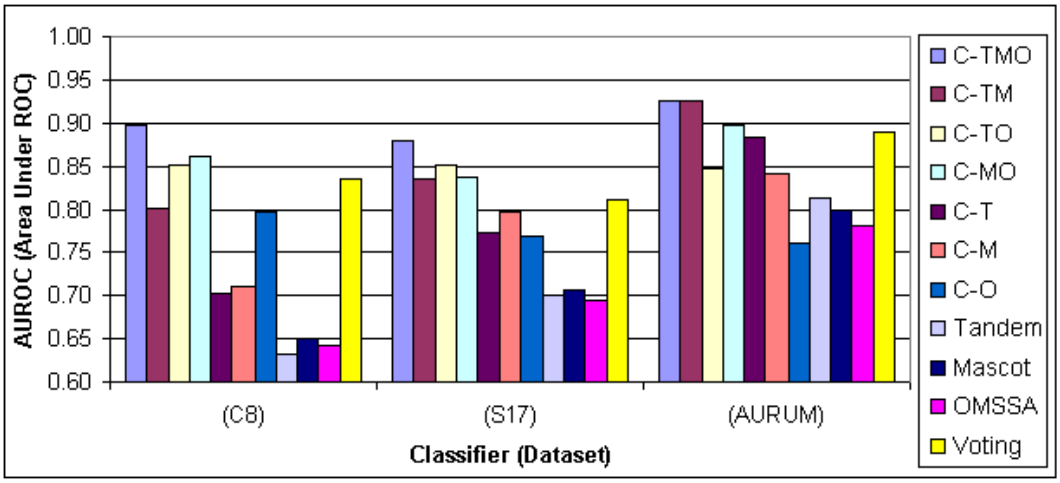

Figure 3.22: AUROC for each classifier and search engine. The y-axis represents the area under the ROC curve (AUROC) for each classifier. Each classifier is displayed as a bar and arranged along the x-axis. Classifiers are arranged in three groups, one for each data set.

the area under the full ROC graph (FPR values between 0 and 1). Results show that when compared using AUROC, C-TMO is more sensitive than all other algorithms for all three data sets.

Sensitivity vs. FDR

Previous ROC graphs measured sensitivity with respect to the false positive rate (FPR). We now examine sensitivity achieved with respect to different values for the false discovery rate (FDR). Because of the differences between FPR and FDR, these results focus on sensitivity for score thresholds corresponding to very small FPR values (on the far left) of ROC graphs.

Figures 3.23, 3.24 and 3.25 show relationship of sensitivity vs. FDR, with six sets of graphs for each data set. In each graph, the y-axis represents sensitivity, and the x-axis represents FDR. The three graphs on the left of each figure present the curves for classifiers C-T, C-M, and C-O (displayed as solid lines) and for search engines Tandem, Mascot, and OMSSA (displayed as dotted lines). The three graphs on the right of each figure present the curves for classifiers C-TM, C-MO, and C-TO (displayed as solid lines). The curve for C-TMO (displayed as dash-dotted lines) is included in all graphs for comparison.

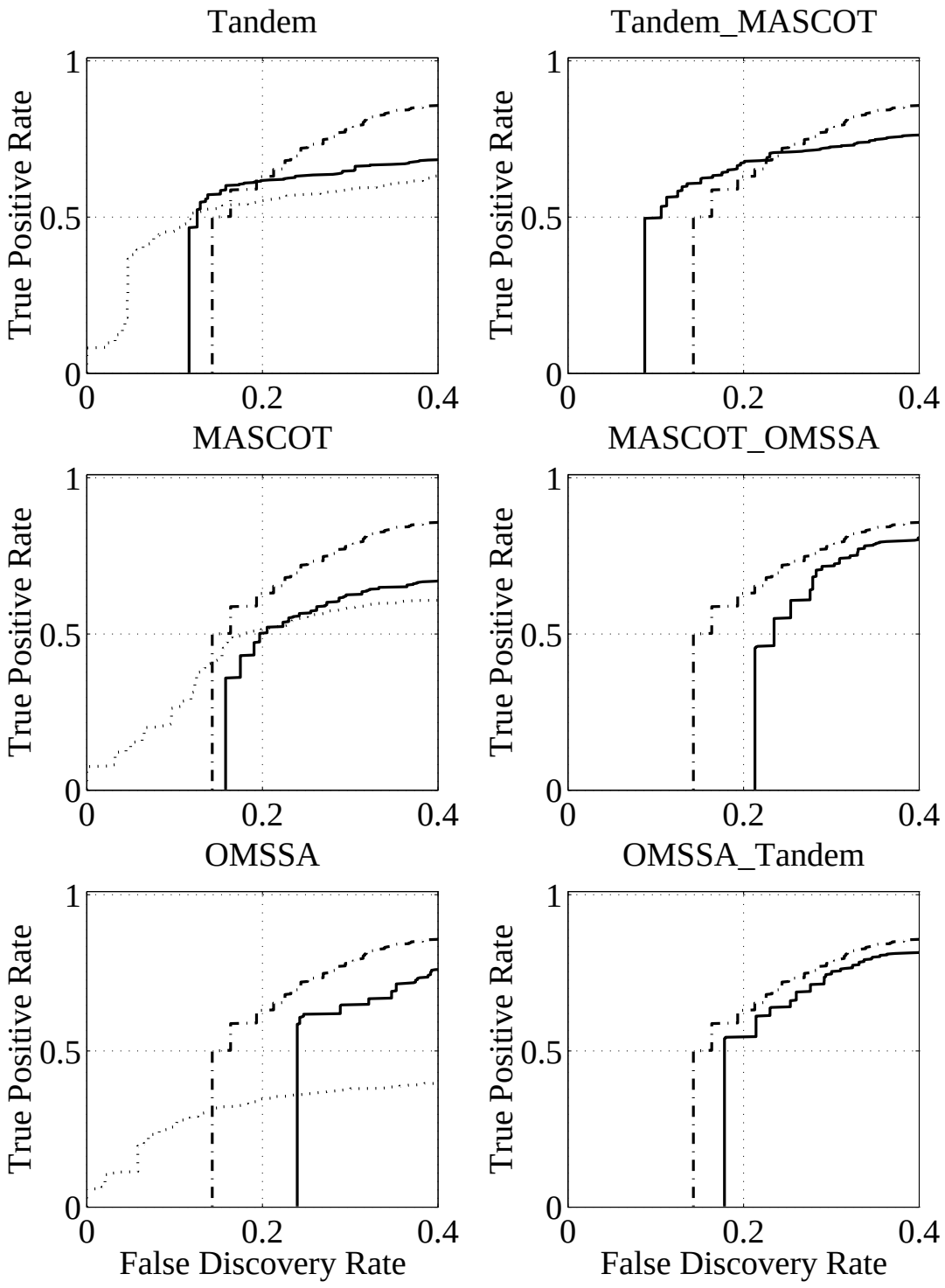

Figure 3.23: Sensitivity vs. FDR curves for C8. Classifiers (solid line) and search engines (dotted line) are presented in each graph. C-TMO classifier (dash-dotted line) is included in all graphs.

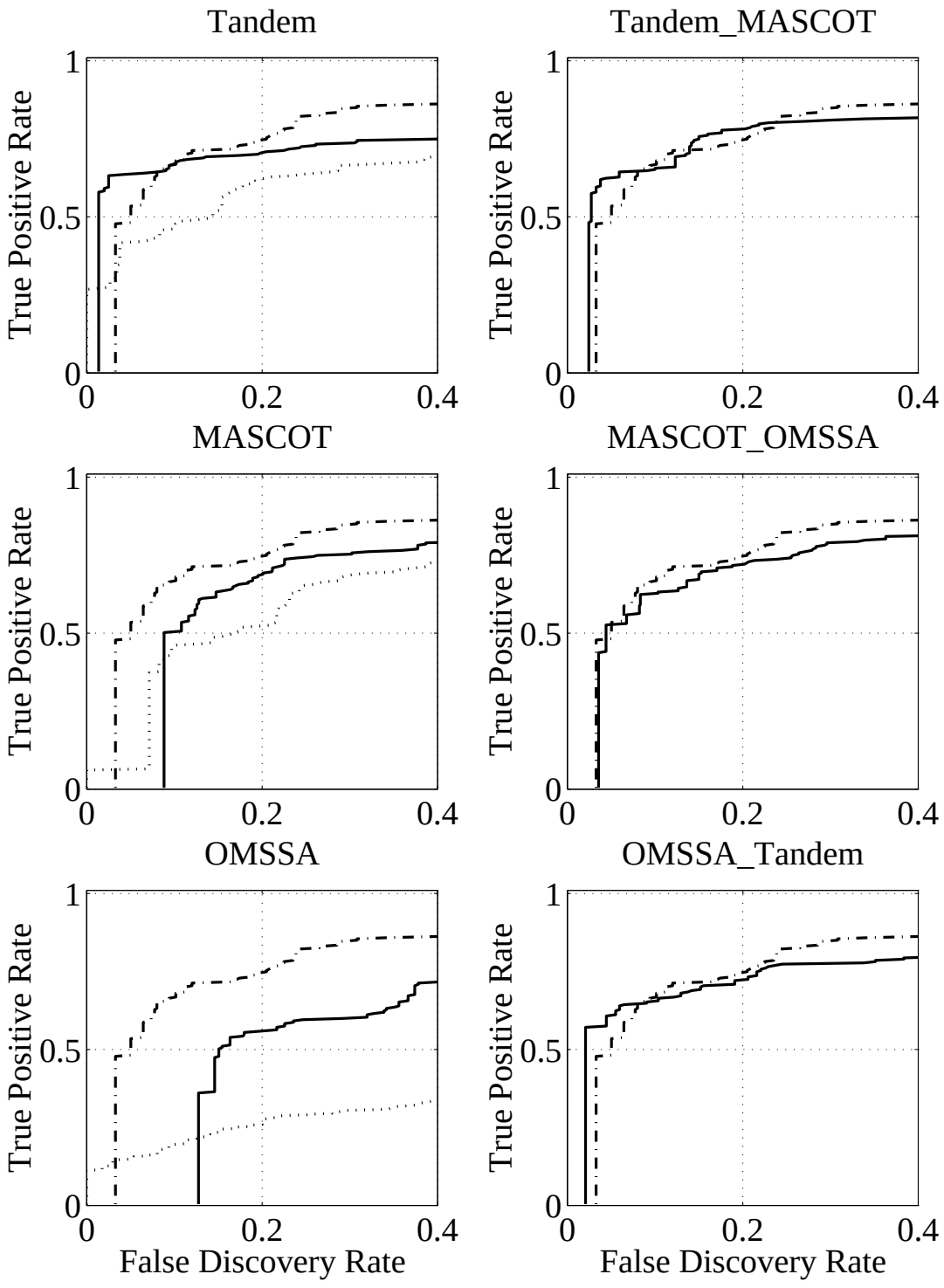

Figure 3.24: Sensitivity vs. FDR curves for S17. Classifiers (solid line) and search engines (dotted line) are presented in each graph. C-TMO classifier (dash-dotted line) is included in all graphs.

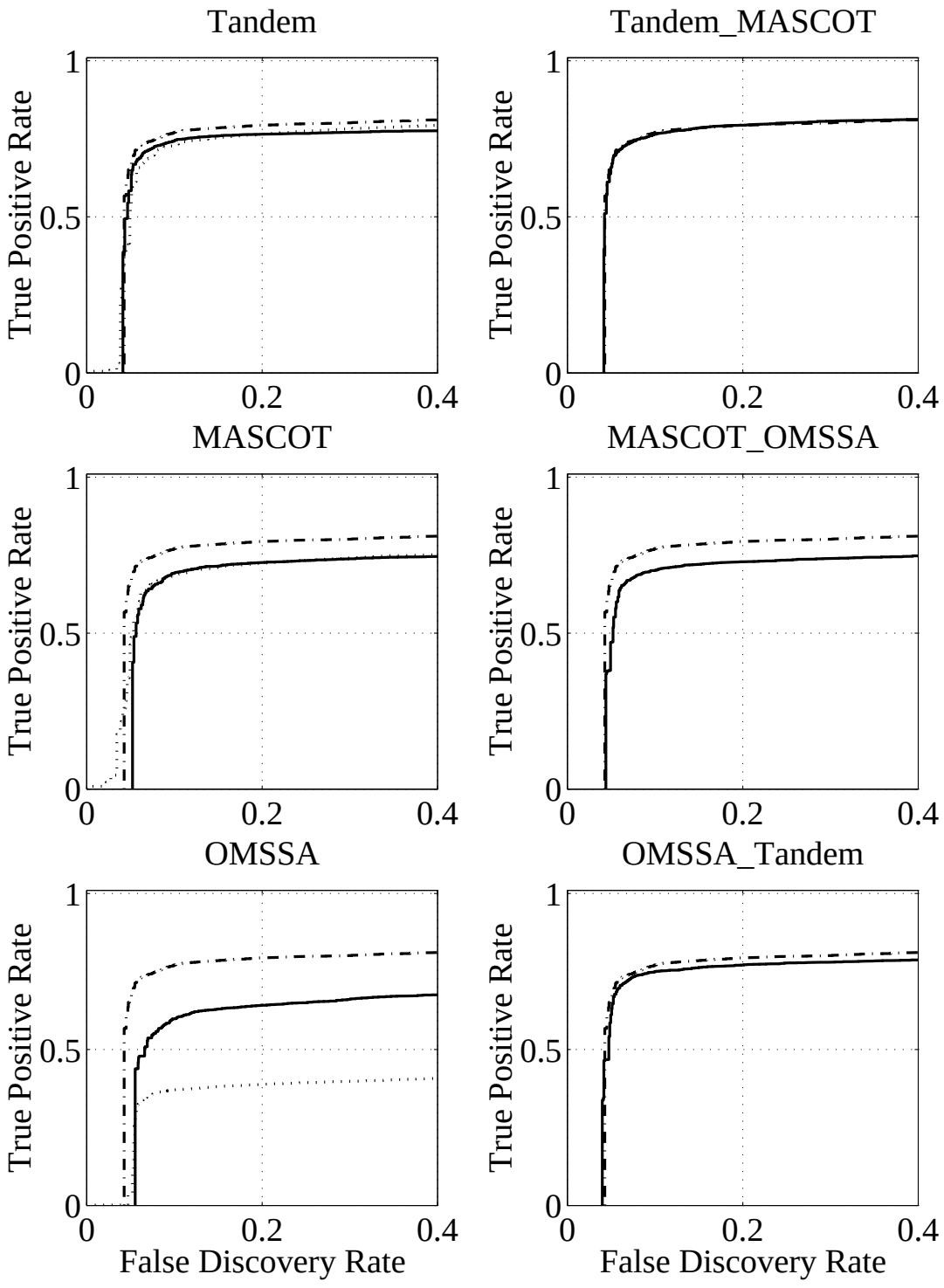

Figure 3.25: Sensitivity vs. FDR curves for AURUM. Classifiers (solid line) and search engines (dotted line) are presented in each graph. C-TMO classifier (dash-dotted line) is included in all graphs.

We see that as with ROC curves, in all three data sets the dash-dotted line (representing C-TMO) is generally above all other lines (demonstrating greater sensitivity). For AURUM C-TMO is almost always best, but for C8 and S17, some Tandem-based classifiers (C-T, C-TO) have better sensitivity at smaller FDR values. These results confirm our earlier conclusions that combining search engine results using machine learning is very sensitive.

A problem that is noticeable in these graphs is that for the C8 and AURUM data sets at very small FDR values, even individual search engines have better sensitivity than any classifier. It turns out that the sensitivity problem with very low FDR values is caused by the underlying Random Forest algorithm employed by classifiers. Setting the threshold at a very low FDR value is equivalent to selecting a very small number of high-confidence peptide IDs. Search engines have no problems with doing so, since they can just pick peptide IDs with very low E-values (e.g., 10^{-60}). Peptide ID scores generated by Random Forest, in comparison, are between 0 and 1.0 based on the fraction of decisions tree voting for the peptide ID. The best score is 1.0, indicating all trees voted for the correctness of the peptide ID.

Since the current PepArML implementation uses 100 trees, with a large number of spectra we will find several peptide IDs with the highest score (1.0). Without using additional trees, PepArML cannot distinguish the "best" peptide ID from this top group. As a result the proportion of true peptide IDs within this top-scoring group establishes the baseline minimum FDR reported in our experiments, preventing the classifier from improving performance for small FDR values.

For instance, assume 20 peptide IDs all receive the highest score from Random

Forest (1.0, indicating unanimous yes votes from all trees). However, only 19 of the peptide IDs are correct, meaning that the lowest FDR possible is 0.05 (1 in 20). In comparison, when using just the search engine score, it is possible to achieve a FDR of 0.0 if the highest scoring peptide ID is correct.

Using larger numbers of trees or other methods to distinguish between these top-scoring, high confidence peptide IDs can ameliorate this problem for Random Forest classifiers.

Sensitivity vs. Estimated FDR

We now examine sensitivity achieved for score thresholds representing different estimated FDR (eFDR) values. Figures 3.26, 3.27 and 3.28 show relationship of sensitivity vs. estimated FDR (eFDR), with six sets of graphs for each data set. In each graph, the y-axis represents sensitivity, and the x-axis represents estimated false discovery rate. The three graphs on the left of each figure present the curves for classifiers C-T, C-M, and C-O (displayed as solid lines) and for search engines Tandem, Mascot, and OMSSA (displayed as dotted lines). The three graphs on the right of each figure present the curves for classifiers C-TM, C-MO, and C-TO (displayed as solid lines). The curve for C-TMO (displayed as dash-dotted lines) is included in all graphs for comparison.

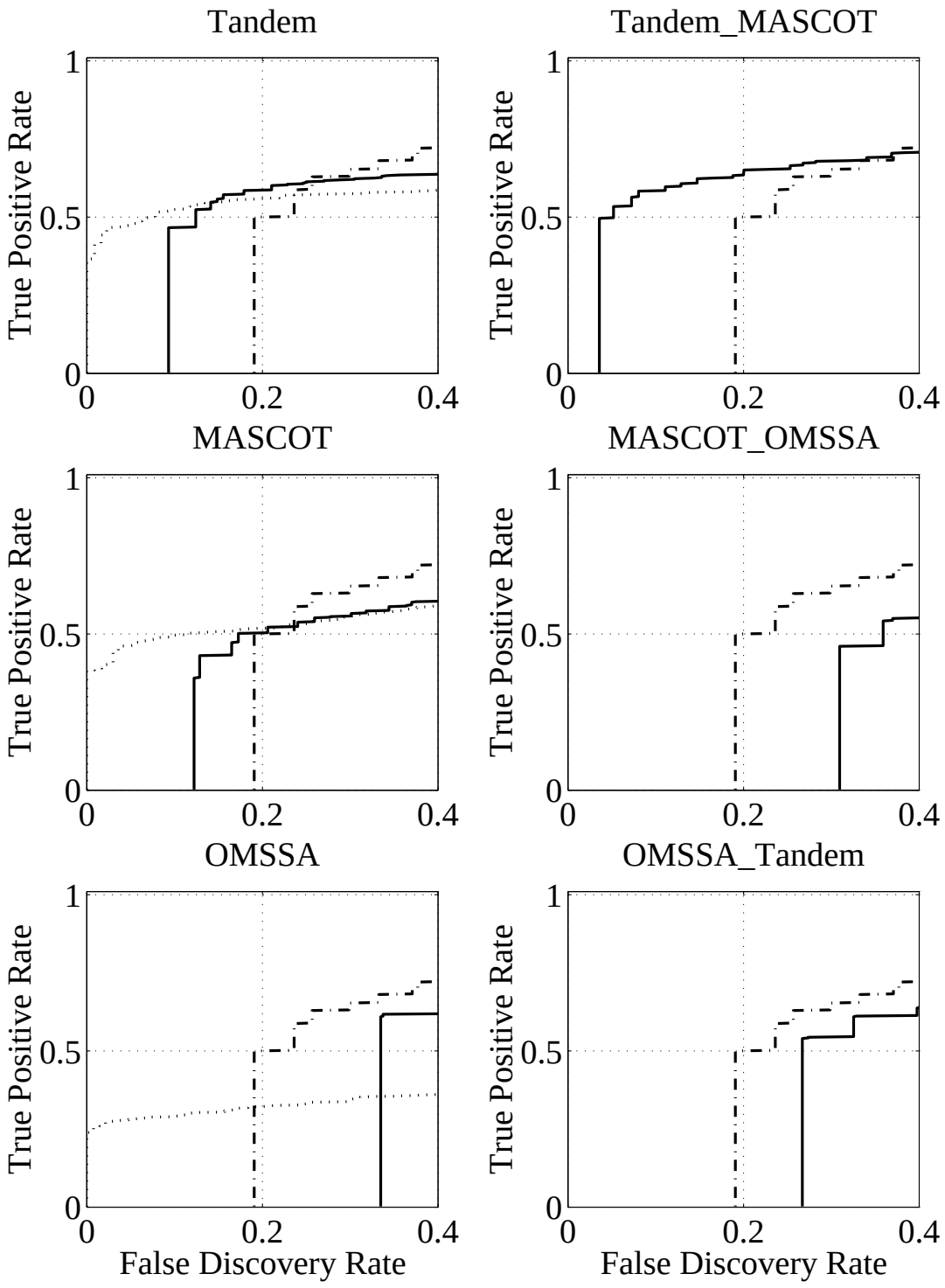

Figure 3.26: Sensitivity vs. eFDR curves for C8. Classifiers (solid line) and search engines (dotted line) are presented in each graph. C-TMO classifier (dash-dotted line) is included in all graphs.

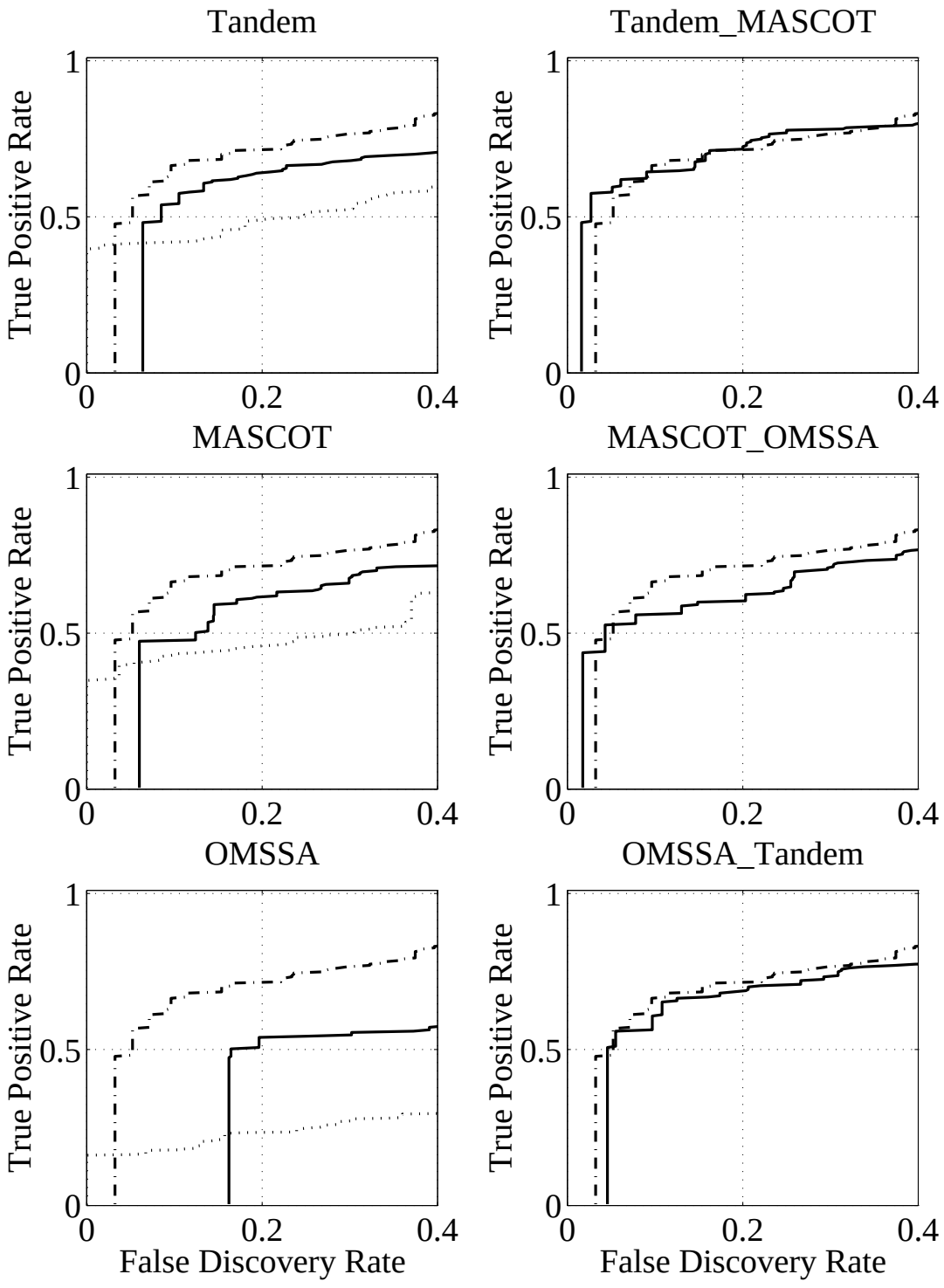

Figure 3.27: Sensitivity vs. eFDR curves for S17. Classifiers (solid line) and search engines (dotted line) are presented in each graph. C-TMO classifier (dash-dotted line) is included in all graphs.

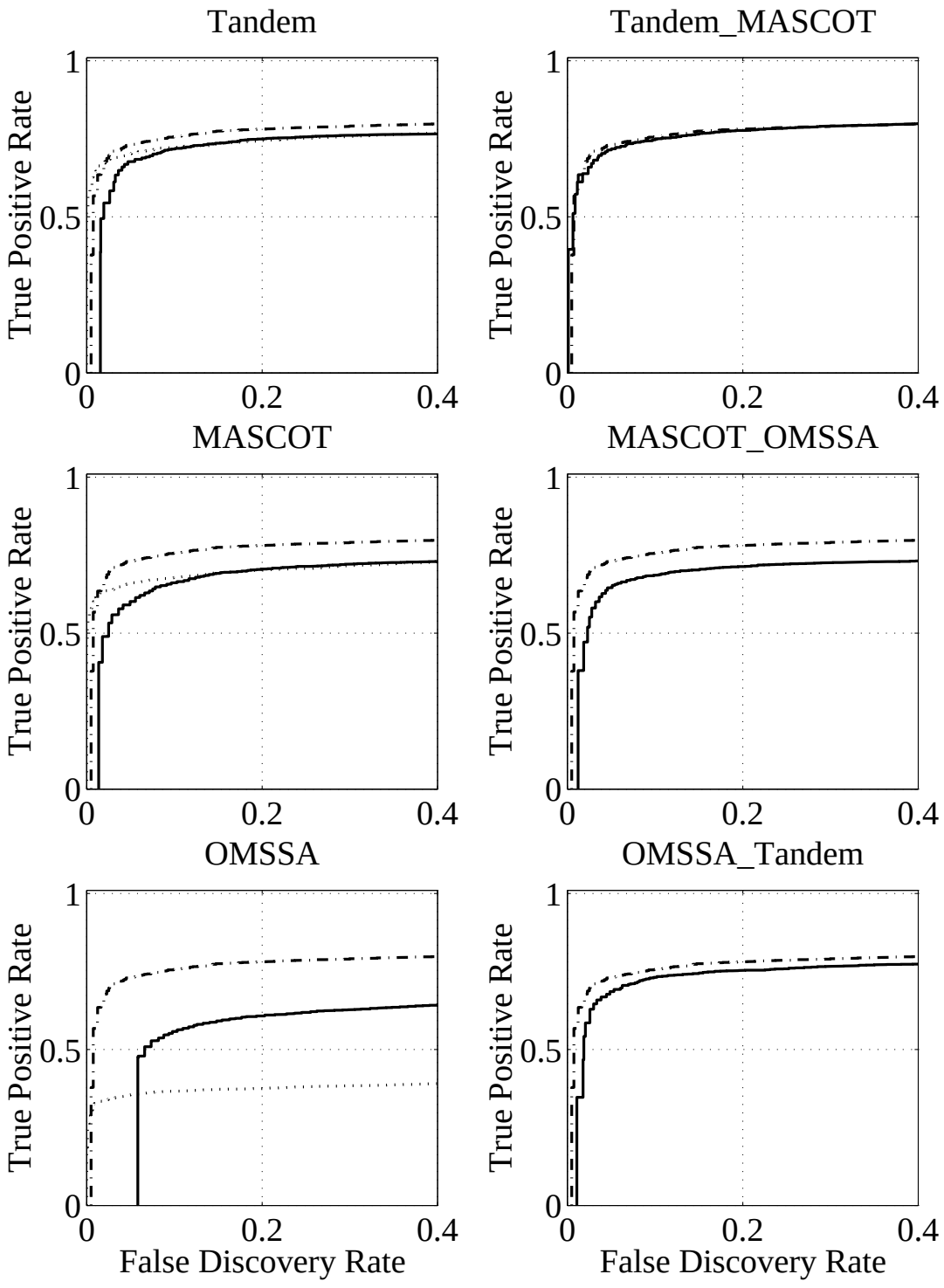

Figure 3.28: Sensitivity vs. eFDR curves for AURUM. Classifiers (solid line) and search engines (dotted line) are presented in each graph. C-TMO classifier (dash-dotted line) is included in all graphs.

Because we can calculate estimated FDR using decoy databases, these graphs indicate what sensitivity can be achieved in practice with real biological samples for different algorithms. We find that unlike ROC curves, the data set chosen has a significant impact on the relative performance of different algorithms. For the AURUM data set C-TMO consistently achieves the best performance. C-TMO performs almost as well for the S17 data set, except for very small eFDR values. However, for the C8 data set C-TMO underperforms both the C-T, C-TM, and C-M classifiers and individual search engines for small values of eFDR.

Note that the actual sensitivity of the classifier remains unchanged. The problem is caused by the difficulty of using eFDR to selecting the appropriate score threshold for the desired sensitivity. These results point out potential difficulties with using eFDR when ground truth is not known.

3.3.8.5 Machine Learning Method Comparisons

We also used Weka to compare different machine learning methods (Random Forest, Logistic Regression, AdaBoost and Naive Bayes) to measure their performance on all three data sets when when combining all three search engine results. Table 3.11 describes the machine learning methods and their default parameters.

The performance of these machine learning methods for combining all three search engines are compared in Figures 3.29, 3.30 and 3.31. Each figure gives the ROC curves of different machine learning methods. Results for SVM on data set AURUM are omitted because of its long running time.

Algorithms	**Description**
Logistic Regression	Build multinomial logistic regression model with a ridge estimator. The log-likelihood of ridge value is 1.0E-8 and maximum number of iterations is -1.
SVM	The polynomial kernel: $K(x, y) =< x, y >^p$ or $K(x, y) = (< x, y > +1)^p$, $p = 4$ John Platt's sequential minimal optimization algorithm for training.
Random Forest	Construct a forest of random trees with unlimited depth of trees and the number of trees to be generated is 10.
AdaBoost	Boost a nominal class classifier DecisionStump using the Adaboost M1 method without resampling. Number of iteration is 10 and weight threshold is 100.
Naive Bayes	Naive Bayesian classifier using estimator classes.

Table 3.11: Machine Learning Classification Algorithms

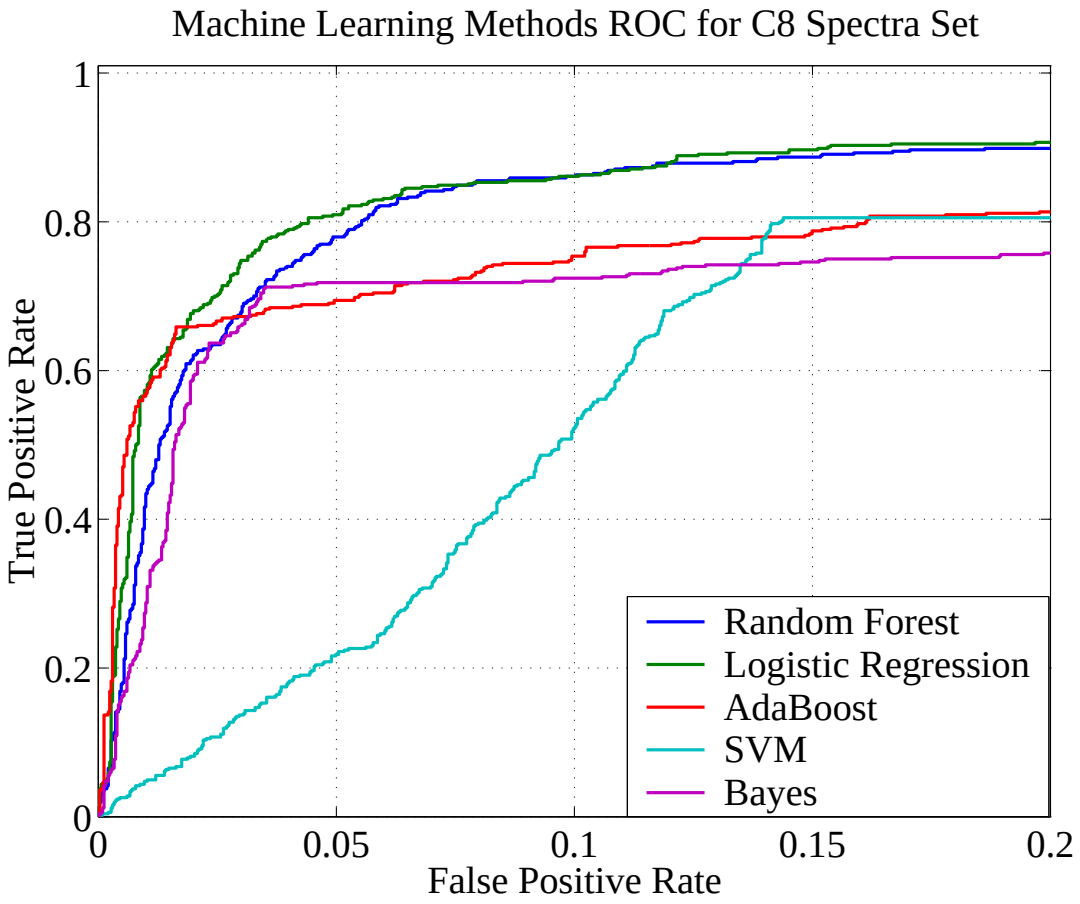

Figure 3.29: Machine Learning Classifiers for C8 Spectra Set

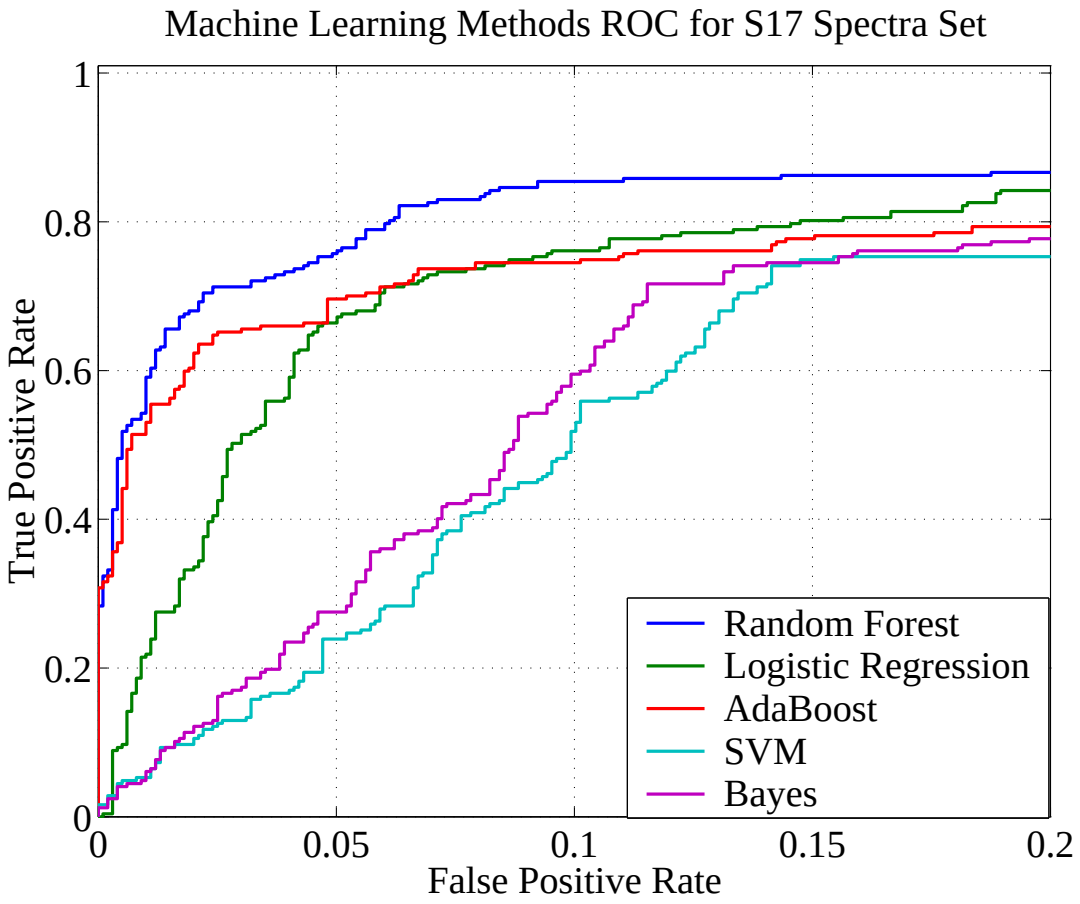

Figure 3.30: Machine Learning Classifiers for S17 Spectra Set

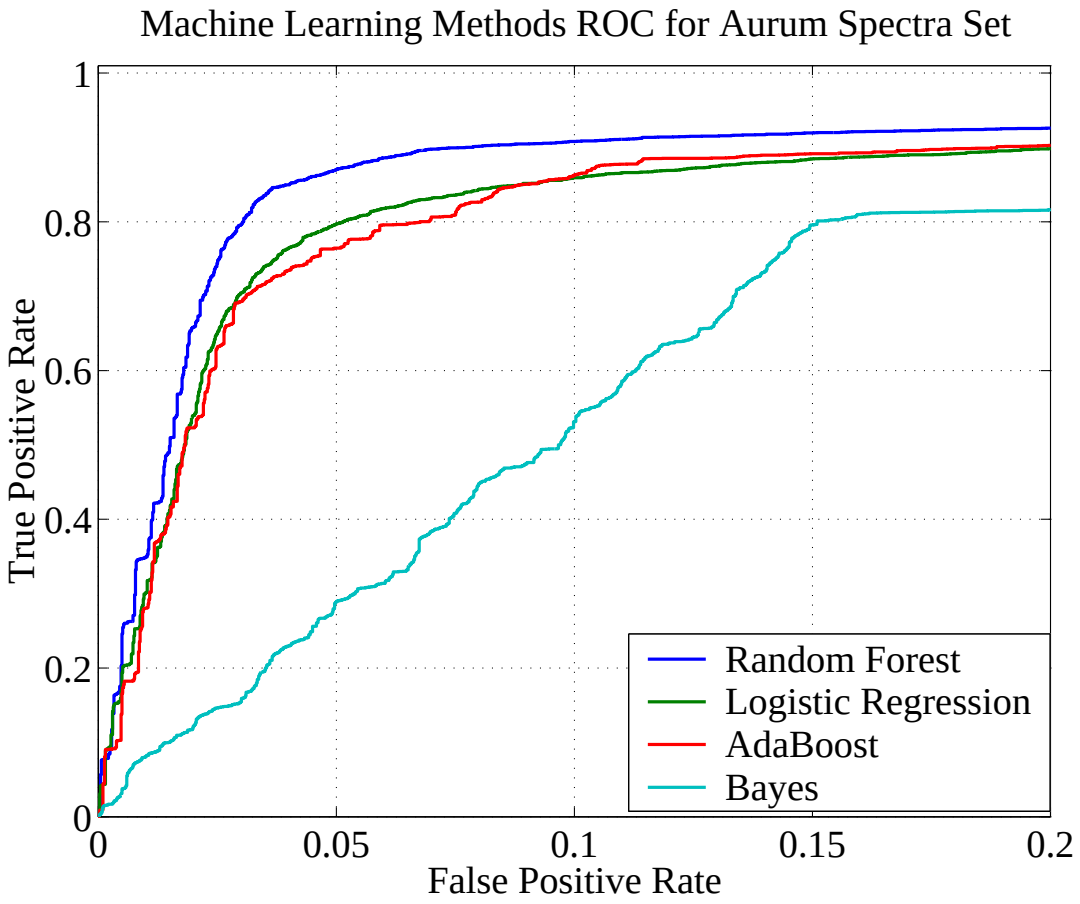

Figure 3.31: Machine Learning Classifiers for AURUM Spectra Set

The performance of Random Forest appears to be best for all data sets, except for smaller FPR values for C8. AdaBoost and Logistic Regression are almost as sensitive, and naive Bayes and SVMs perform poorly. It is possible SVM performance can be improved with a higher order polynomial kernel, but would be even more computationally intensive.

3.3.8.6 InfoGain For Machine Learning Features

We assess the discriminating ability of each element of the feature vector using the information gain metric. The information gain of a feature measures the reduction in entropy or randomness when the data set is subdivided according to the feature's values. Information gain, called *InfoGain* by *Weka*, is one of many useful techniques for assessing the relative importance of each feature in machine learning classifiers, and is a property of the data set, independent of the particular classification algorithm used. Larger InfoGain values indicate a feature is likely to be more useful in making accurate predictions.

The InfoGain values of search engine sentinels are of particular interest. Search engine sentinels have value 0 or 1 depending on whether the search engine result contains a particular peptide-spectrum assignment as a top-ranked peptide ID. Search engine features, which take sentinel values when missing, also provide this information, but the significance of the sentinel values is masked by the real feature values. The InfoGain values for sentinels can therefore help gage the benefit of the additional search engine features; features need to achieve larger InfoGain than the

sentinels to demonstrate benefit over voting or consensus peptides.

Table 3.12 and Figure 3.32 shows InfoGain values computed by Weka for each feature of the feature vector. We see that sentinels provide low InfoGain scores, indicating that whether a search engine managed to make any prediction for a spectrum to be not particularly useful. Similarly, the low InfoGain score for peptide length indicates it is not a useful predictor of correctness. *E*-values and raw search engine scores have among the highest InfoGain scores and thus provide important information. Number of ions matched is also useful. Surprisingly, precursor mass delta (the difference between expected vs. measured precursor mass) is a good indicator.

3.3.8.7 Generality of Machine Learning Model

Table 3.12 shows large variation in InfoGain and relative rank for each feature between data sets. This suggests that a classifier model trained on one data set may perform poorly when applied to another. To demonstrate this effect, we applied the classifier trained on the C8 data set to the S17 data set, and compared its sensitivity to a classifier actually trained on the S17 data set. The performance comparison between two models are shown by ROC curves in Figure 3.33.

Search Engine	Feature	C8	S17	AURUM
—	Peptide length	0.12 (14)	0.06 (20)	0.00 (20)
Tandem	Hyperscore	0.14 (9)	0.33 (6)	0.55 (2)
	Precursor mass delta	0.13 (12)	0.31 (9)	0.47 (6)
	# of matched y-ions	0.10 (16)	0.30 (10)	0.47 (7)
	# of matched b-ions	0.07 (18)	0.19 (14)	0.36 (12)
	# of missed cleavages	0.14 (11)	0.23 (13)	0.31 (13)
	Sum of matched intensity	0.15 (8)	0.17 (16)	0.29 (14)
	E-value	0.28 (5)	0.38 (3)	0.58 (1)
	Sentinel	0.07 (19)	0.16 (17)	0.23 (16)
Mascot	Score	0.30 (3)	0.40 (2)	0.51 (4)
	Precursor mass delta	0.12 (15)	0.33 (5)	0.43 (9)
	# of matched ions	0.17 (6)	0.37 (4)	0.42 (11)
	# of matched peaks	0.13 (13)	0.24 (12)	0.22 (17)
	# of missed cleavages	0.14 (10)	0.18 (15)	0.26 (15)
	E-value	0.28 (4)	0.40 (1)	0.52 (3)
	Sentinel	0.06 (20)	0.10 (18)	0.19 (18)
OMSSA	p-value	0.34 (2)	0.32 (7)	0.47 (8)
	# of matched ions	0.17 (7)	0.27 (11)	0.42 (10)
	E-value	0.34 (1)	0.32 (8)	0.47 (5)
	Sentinel	0.10 (17)	0.10 (19)	0.13 (19)

Table 3.12: InfoGain (Rank) of features with respect to each data set.

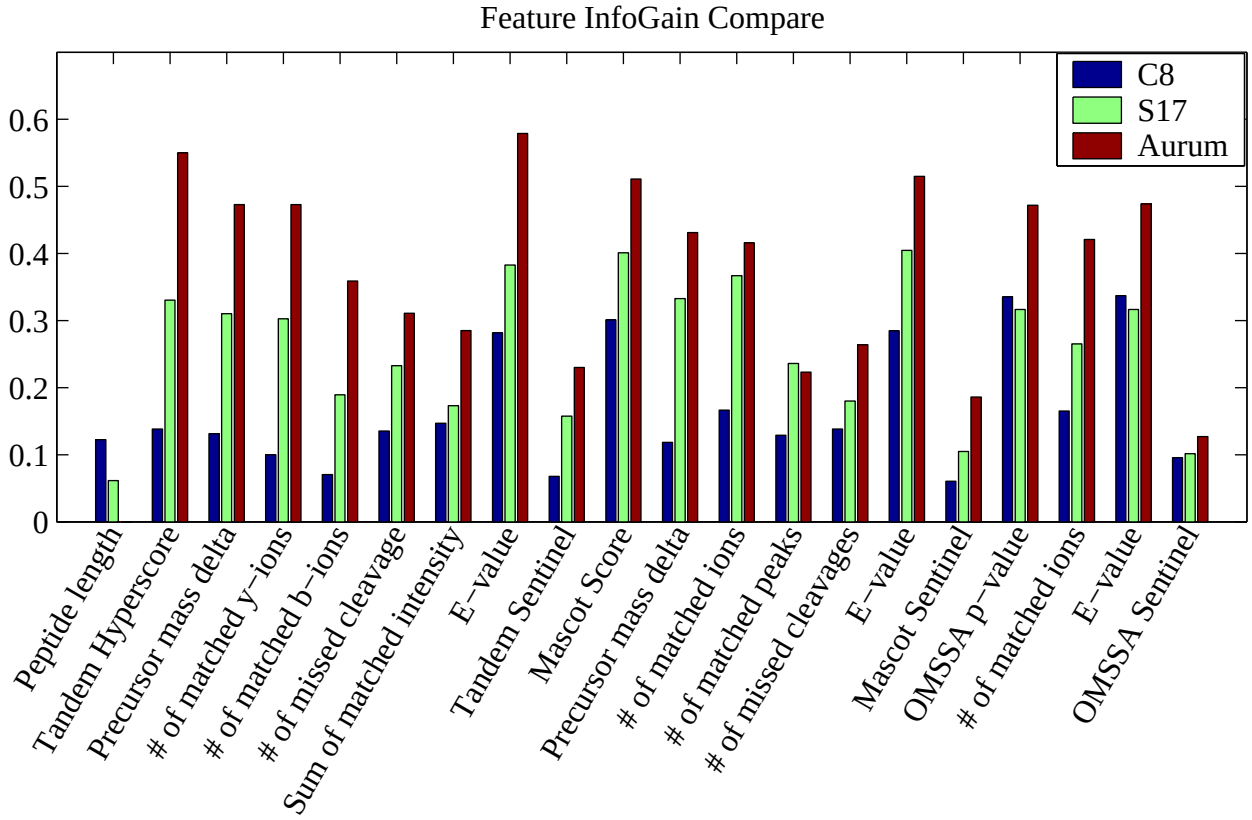

Figure 3.32: InfoGain For Features

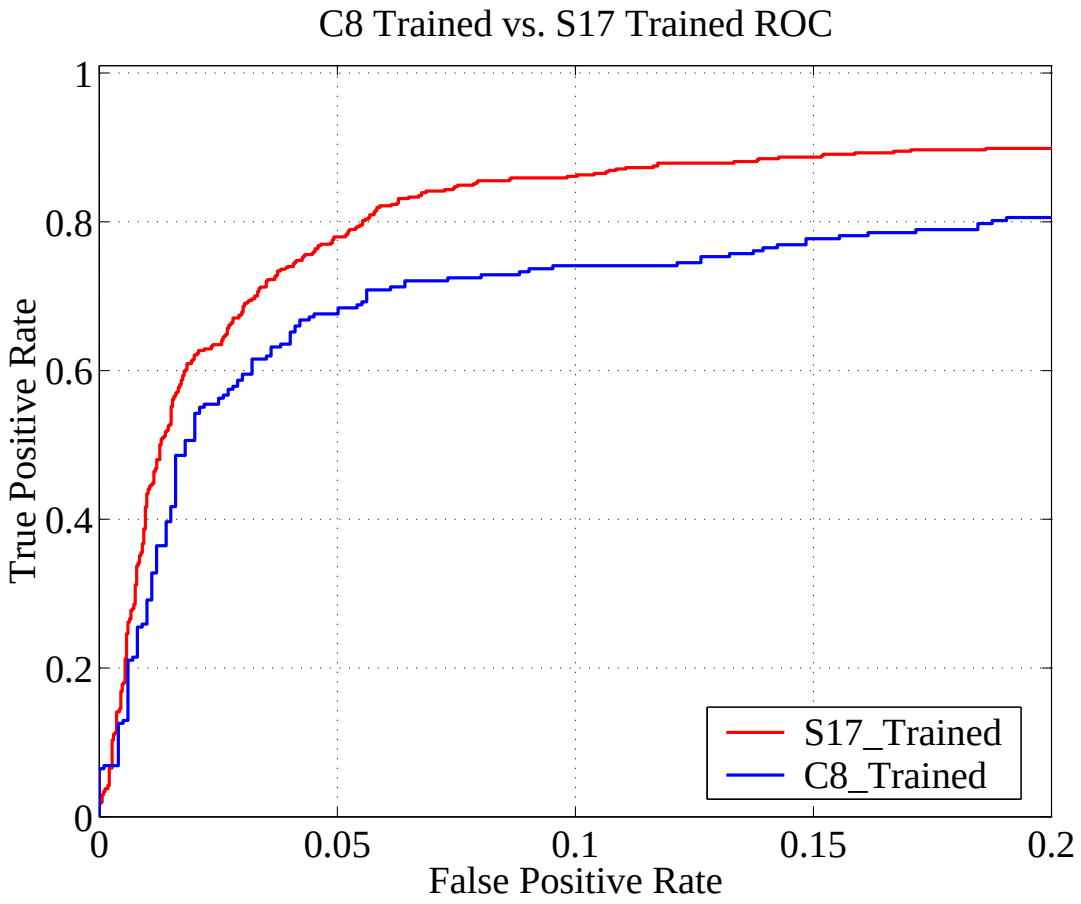

Figure 3.33: ROC curves for C-TMO trained with S17 or C8, both applied to S17.

The C8 classifier appears to be not very accurate when used to classify spectra in the S17 data set, performing worse than the S17 classifier. This difference may have many causes, ranging from differences in protein content, sample preparation, mass spectrometers, search engine parameters, number of mass spectra, percentage of true positive spectra, etc.

In general, when applying a classifier to a completely different data set, not only could the machine learning algorithms choose sub-optimal features, but the weights and thresholds estimated by training would be invalid if the characteristics of the feature values changed too much. For example, E-values change with sequence database size and precursor mass tolerance, and search engine scores tend to depend heavily on the instrument fragment mass tolerance. We thus believe it is difficult to generalize machine learning models trained on one data set for new data sets.

3.3.8.8 Unsupervised Machine Learning Performance

Since it appears difficult to construct generalized classifiers for use on a range of data sets, for machine learning tools to be useful in practice, models must be able to be automatically trained on the specific characteristics of each data set in an unsupervised fashion. The unsupervised PepArML training procedure is designed to be able to automatically predict correct labels for peptide IDs, making it possible to train the classifier without supervision.

Section 3.3.5.3 describes the PepArML unsupervised training approach in detail. Table 3.13 lists the actual parameters used by the current PepArML implemen-

	Initial Parameter		Iterative Run Parameter		
	Non-overlap peptides	**Protein Coverage**	**eFDR**	**Non-overlap peptides**	**Protein Coverage**
C8	≥ 2	≥ 0.3	≤ 0.2	≥ 2	≥ 0.1
S17	≥ 2	≥ 0.1	≤ 0.2	≥ 2	≥ 0.1
AURUM	≥ 2	≥ 0.1	≤ 0.5	≥ 2	≥ 0.1

Table 3.13: Parameters for Unsupervised Learning

tation of unsupervised training for each data set. Initially, only peptide IDs agreed on by all three search engines are used to calculate the number of non-overlapping peptides and protein coverage percentage. After the first classifier is trained, only peptides with eFDR values below 0.2 (for C8 and S17) or 0.5 (for AURUM) are used to calculate the number of non-overlapping peptides and protein coverage percentage.

To demonstrate the effectiveness of the unsupervised PepArML training procedure, we compare its sensitivity to the supervised version of PepArML. The left side of Figure 3.34 shows the ROC curves of supervised (dotted line) and unsupervised (solid line) learning C-TMO classifiers applied to each data set.

These results clearly demonstrate the sensitivity of the unsupervised PepArML classifier, since little is lost by using the heuristic unsupervised training procedure, which is carried out without knowledge of the true proteins and peptide IDs.

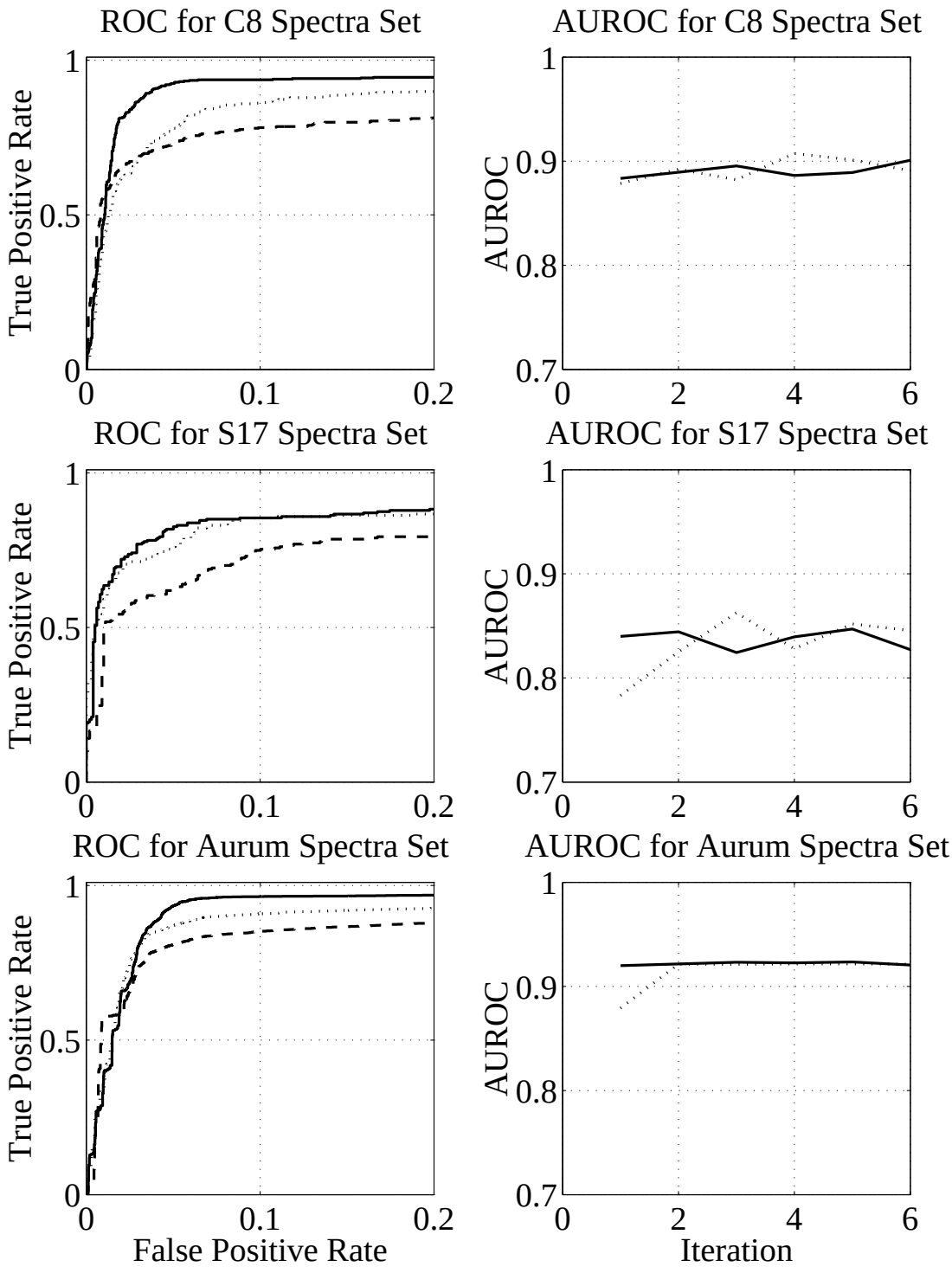

Figure 3.34: Left: supervised C-TMO (dotted), unsupervised C-TMO (solid), and Voting (dashed). Right: unsupervised C-TMO starting with consensus proteins (solid), or starting with single random true protein (dotted).

	C8		S17		AURUM	
Classifiers	**10% FDR**	**20% FDR**	**10% FDR**	**20% FDR**	**10% FDR**	**20% FDR**
Unsup	0.00	**0.89**	**0.70**	**0.83**	**0.83**	**0.83**
C-TMO	0.00	0.63	0.67	0.75	0.77	0.79
C-TM	**0.50**	0.68	0.65	0.78	0.76	0.79
C-TO	0.00	0.55	0.66	0.72	0.75	0.77
C-MO	0.00	0.00	0.63	0.72	0.70	0.73
C-T	0.00	0.62	0.67	0.71	0.74	0.77
C-M	0.00	0.50	0.51	0.69	0.69	0.73
C-O	0.00	0.00	0.00	0.56	0.60	0.64
Tandem	0.46	0.55	0.46	0.62	0.73	0.77
Mascot	0.26	0.52	0.46	0.53	0.69	0.72
OMSSA	0.26	0.35	0.20	0.26	0.37	0.39

Table 3.14: Sensitivity vs. true FDR for supervised & unsupervised classifiers. Best sensitivity for each FDR & data set in bold.

	C8		S17		AURUM	
Classifiers	**10% eFDR**	**20% eFDR**	**10% eFDR**	**20% eFDR**	**10% eFDR**	**20% eFDR**
Unsup	**0.69**	**0.84**	0.60	0.69	0.82	0.83
C-TMO	0.00	0.50	**0.67**	**0.72**	0.76	0.78
C-TM	0.59	0.64	0.65	**0.72**	0.73	0.75
C-TO	0.00	0.00	0.61	0.69	**0.85**	**0.94**
C-MO	0.00	0.00	0.56	0.60	0.69	0.71
C-T	0.47	0.59	0.54	0.64	0.72	0.75
C-M	0.00	0.50	0.47	0.62	0.66	0.71
C-O	0.00	0.00	0.00	0.54	0.56	0.61
Tandem	0.52	0.56	0.42	0.49	0.72	0.74
Mascot	0.50	0.52	0.43	0.46	0.68	0.70
OMSSA	0.29	0.32	0.18	0.23	0.37	0.37

Table 3.15: Sensitivity vs. estimated FDR for supervised & unsupervised classifiers. Best sensitivity for each eFDR & data set in bold.

Tables 3.14 and 3.15 present the sensitivity of the unsupervised learning PepArML classifier compared to other algorithms for at different FDR and eFDR values.

While the performance evaluation results demonstrate that the unsupervised learning PepArML procedure achieves similar sensitivity as the supervised learning approach on all three data sets, its rate of convergence and tolerance for a poor initial putative true protein set still needs to be established. If too many iterations are required, or if small errors in initial protein set can result in large number of nonsense predictions, then whether the technique can be applied to real experimental data sets may be in doubt.

The graphs in the right column of Figure 3.34 show the convergence rate of the unsupervised PepArML training procedure. The x-axis represents the training iteration of the unsupervised classifier. The y-axis represents the AUROC of the classifier after each iteration of training, evaluated with respect to the current putative true protein set. The solid line shows the convergence rate for the C-TMO classifier using an initial putative true protein set selected based on the PepArML algorithm (i.e., peptide IDs agreed on by all three search engines). The AUROC of the classifier converges quickly, stabilizing within three iterations of training.

The robustness of the unsupervised learning PepArML is also tested by choosing more challenging initial true positive protein sets representative of various types of potential errors. The dotted lines in the graphs in the right column of Figure 3.34 show the rate of convergence when the initial putative true protein set consists of just one randomly selected true positive protein for C8 and S17 data set (and ten

randomly selected true positive proteins for AURUM data set). These cases represent starting from very conservative assumptions, where many true positive proteins are falsely labeled as incorrect.

Despite many incorrectly labeled peptide IDs, with the PepArML unsupervised training algorithm convergence of AUROC is quick and shows little evidence of instability. This is particularly impressive for the AURUM data set which contains hundreds of true positive proteins and for which the unsupervised learning PepArML stabilizes after only three iterations. For data sets with a small number of true positive proteins (C8), convergence is almost as quick as when using the full set of putative true proteins. Similar quick convergence is also achieved with other difficult initial putative true protein sets. As a result, we believe that unsupervised PepArML is quite robust with respect to the choice of initial putative true protein set, at least for synthetic protein mixes.

3.3.8.9 Discussion of Experimental Results

To recap, our experimental evaluation of the sensitivity and selectivity of individual search engines, combining heuristics, and machine learning classifiers provided considerable insight on their relative performance. Our major observations were:

Comparing peptide identifications using estimated FDR (eFDR) is more effective than comparisons using E-values. Results show eFDR is more closely correlated to true FDR than search engine E-values, and heuristic combiners based on eFDR

are more sensitive than combiners based on E-values.

Heuristic combiners can improve on single search engine scores. Improvement in sensitivity based on The best-performing heuristic combiners demonstrate clear improvements in sensitivity over individual search engines for all three data sets tested. If a machine learning framework such as PepArML cannot be readily applied, implementing an eFDR-based voting heuristic combiner is still worthwhile.

Machine learning classifiers outperform single search engine scores. By exploiting information in additional features, classifiers can significantly improve sensitivity beyond search engine scores, particularly when search engines provide many additional useful features.

Classifiers based on multiple search engines outperform voting heuristic combiners. The ability of machine learning classifiers to achieve greater sensitivity when combining search engine results is likely due to a combination of more accurate modeling of individual search engines, and ability to use non-score features.

Classifiers based on multiple search engines outperform classifiers based on features from a single search engine. Classifiers generally benefit from additional information. The best overall performance is generally achieved by C-TMO, the classifier trained using data from all three search engines. It is unclear whether the performance boost is due to complementary properties of features from different search engines, or the incorporation of agreeing peptide ID information in the feature vec-

tor.

Adding features does not guarantee improved classifier performance. While additional features generally improve classifier performance, some features may provide poor discrimination and interfere with machine learning, degrading performance. In other cases, additional features may be highly correlated with existing features and the benefit of additional peptide ID agreement is minimal.

Combining two search engines may be sufficient. Even though C-TMO usually yields the most accurate results, classifiers constructed using only data from two search engines (C-TM. C-MO, C-TO) often achieve similar sensitivity. This result is particularly interesting from a commercial perspective, as C-TO, a classifier built using only open source software, can approach the sensitivity of classifiers (C-TMO, C-TM) that utilize features from Mascot, a commercial search engine. This suggests that the machine learning combination of open-source search engines may be a viable alternative to commercial search engines.

Unsupervised learning works well. Overall, we observe that the sensitivity of unsupervised classifiers is very similar to that of supervised classifiers, sometimes even better. Unsupervised PepArML performs well even for the AURUM data set, which contains hundreds of proteins.

Applicability for biological samples. Our current success has been achieved with data sets derived from synthetic protein mixes. Real biological samples are likely

to be more complicated, and may require more robust algorithms, particularly for unsupervised learning. Nonetheless, we believe PepArML demonstrates the potential for using machine learning to automatically combine search engine results and features.

3.3.9 Conclusion

A highly sensitive and specific new technique is demonstrated here for separating true from false peptide identifications on three synthetic protein mixture data sets from both electrospray and MALDI instruments. PepArML uses machine learning to combine the search results from many search engines, achieving better results than machine-learning or result combining alone. PepArML uses the model-free random forest machine learning technique, which ensures that the relative contributions of search engine agreement and peptide-spectrum match scores can be used optimally for each combination of spectra, search engines, and search engine parameters. It is shown that PepArML can be trained effectively in an unsupervised manner, making it possible to removes the need for extensive libraries of pre-trained models based on experimental spectra from synthetic protein mixtures from all instrument, search engine, and parameter combinations. Unsupervised PepArML training also alleviates concerns about sub-optimal machine learning models being applied beyond their ability to generalize effectively.

PepArML does not rely on specialized features or difficult to compute scores, but these can be easily added to the model if desired. Similarly, PepArML can be

used with any number of different search engines, or even multiple searches from the same search engine. The model-free nature of PepArML combining even makes it possible to combine results from disparate peptide identification techniques, such as spectral matching, alongside search engine results, or to use paired searches with conservative and aggressive search parameters.

The unsupervised training procedure could be manipulated in a variety of ways to exercise more control over PepArML learning. Users could hand select the initial set of putative true proteins, or apply a species constraint to the putative true protein set, if the sample is known to come from a particular organism. It would also be straightforward to incorporate peptide IDs to decoy peptides as known false identifications just as other semi-supervised learning approaches have done. However, the addition of known false peptide IDs is less powerful in this context than correctly guessing true labels on a smaller number of spectra.

The excellent performance of PepArML applied to the results of Tandem and OMSSA, both open-source, freely available search engines, raises the tantalizing possibility that a PepArML based meta-search-engine might offer superior identification performance than costly commercial search engines. Such a meta-search-engine could wrap Tandem, OMSSA, and other free search engines behind a single user interface.

It remains to be seen whether the iterated unsupervised learning procedure proposed here can be applied, as described, across the rich variety of experimental data sets. Our experiments to investigate the robustness of the procedure are encouraging, but it is possible our simple technique for selecting putative true pro-

teins, in particular, may find too few true positive proteins for successful training. In future work, the plan is to investigate whether protein identification tools, such as Protein Prophet [NKKA03], or an expectation-maximization (E-M) approach, which would fit naturally into the iterative PepArML framework, might provide additional robustness for PepArML when applied to experimental data sets.

The algorithm design and experimental results show that the proposed PepArML machine-learning framework has the potential to solve the critical problems of combining multiple search engine results and the use of machine-learning tools beyond training data sets, resulting in robust, effective and reliable tools for peptide spectrum assignment.

Chapter 4

Conclusion and Future Work

4.1 Two Classes of Algorithms Demonstrate Good Performance and Precision

In this thesis, three algorithms in the areas of genomics and proteomics were designed, implemented and evaluated to demonstrate that applying high performance computing techniques and statistical machine learning techniques can improve both the performance and precision of bioinformatics algorithms. In addition to developing several novel techniques, this thesis has performed extensive experimental evaluation significantly beyond the scope of previously published research.

4.1.1 Genomics

In area of genomics, ESTmapper, a DNA to genome alignment algorithm was designed and implemented to efficiently align cDNA sequences to genome(s) with high speed and precision. It also provides flexibility of aligning cDNA sequences to multiple chromosomes with high speed and accuracy. Experiment evaluation results show that ESTmapper has comparable global alignment precision, and at least 2–3 times faster compared to previous algorithms. With a more accurate splice site model, ESTmapper can be used to find alternative splicing isoforms,

cluster EST/cDNA sequences and help with gene finding.

The main contributions of ESTmapper are demonstrating the feasibility of using suffix trees for entire genomes, techniques for estimating the statistical significance of DNA to genome alignments.

4.1.2 Proteomics

In the area of proteomics, two different algorithms for mass spectra based peptide identification are designed and implemented to improve peptide identification accuracy.

4.1.2.1 HMMatch

HMMatch was designed to use hidden Markov model to capture the mass spectra peak intensity consensus and variation pattern. Each model summarizes many examples of a peptide's fragmentation spectrum in a generative probabilistic model. The unassigned mass spectrum can be compared with HMMs to find its peptide identification. Our preliminary experiment results show that HMMatch has good specificity and superior sensitivity, compared to sequence database search engines such as X!Tandem. As the size of publicly available mass spectra databases continues to grow, it is possible to build a library of HMMs for peptide fragmentation pattern. A relatively complete HMM mass spectra library can be a very good complement to current database search based peptide identification methods.

The main contributions of HMMmatch are techniques for calculating statistical

significance of HMM scores, and model extrapolation to peptides without experimental mass spectra.

4.1.2.2 PepArML

PepArML is a machine learning based, model-free framework for peptide identification. It uses the Random Forest method to combine peptide identification results from multiple peptide identification tools. Multiple matrices that compare the similarity between theoretical and experimental spectra are combined under the framework to distinguish true and false peptide identifications. An unsupervised training approach makes it possible for PepArML to learn the properties of each data set on the fly, removing the need to build a comprehensive library of pre-built models based on instrument, search engine, and search parameter combinations. Our preliminary results based on three search engines (X!Tandem, MASCOT, OMSSA) and three synthetic protein mixture data sets from both electrospray and MALDI instruments show the machine learning based framework outperforms machine learning techniques applied to a single search engine's output.

The main contributions of PepArML are demonstrating the effectiveness of machine learning for combining features and scores from multiple search engines, and techniques for unsupervised training for unlabeled mass spectra.

4.1.3 Benefits of High Performance Computing and Machine Learning

Both high performance computing and machine learning techniques have been widely applied to solving problems in bioinformatics field.

Currently, many bioinformatics problems (such as microarray gene expression data analysis, genetic network, protein-protein interactions, phylogeny reconstruction, protein structure predictions, etc.) require computationally intensive operation on a large data domain. It is impractical to solve these problems with only traditional sequential algorithms. Here both parallel computing techniques and carefully designed data structure and algorithms are needed to save data processing time. So high performance computing plays a very important role in solving these problems. In this thesis, a novel sequence alignment algorithm is presented to demonstrate the power of high performance computing techniques by aligning millions of cDNA sequences to genomes within hours. The algorithm not only uses multiple processors on both shared memory and distributed memory architecture, but also utilize suffix tree data structure's linear search speed for string match to speed up each cDNA to genome alignment procedure. The results shows at least 2 - 3 times speed up compared with other current popular sequence alignment softwares.

Machine learning techniques play key role in all bioinformatics research areas, especially in genomics and proteomics areas. In this thesis, two machine learning frameworks are designed and implemented to demonstrate machine learning's application in tandem mass spectra based peptide identification. HMMatch uses hidden

Markov model's probabilistic strength to capture the mass spectra peak intensity consensus and variation pattern for spectral matching based peptide recognition. It show superior sensitivity and specificity compared with current peptide identification methods. PepArML fully utilize the properties of mass spectra data and random forest algorithm to design a unsupervised machine learning framework for combining peptide identification results from multiple search engines. The machine learning framework provides a more accurate method for classifying true from false peptide identifications and improve peptide identification results.

4.2 Future Work

4.2.1 ESTmapper

ESTmapper is an efficient algorithm to align cDNA sequences to genome with high accuracy and speed. It can help with identifying genome structure and provide more accurate splice site information for training gene finding algorithms. Splice site discovery is one of the most important issues in computational gene finding. Some well-designed models have been built to predict splice sites. However, with only a limited amount of accurate exon/intron boundary data available, most current program use consensus sequences to predict splice sites. ESTmapper can be adapted to include more accurate splice site model and branch site model to improve the alignment accuracy. The improved ESTmapper can be used to identify a large number of exon/intron boundaries, which can be used as training data for creating more accurate splice site model and gene finders. It can also be used to detect

alternative splicing isoforms and micro-exon, which is a challenging problem in the field of bioinformatics.

4.2.2 HMMatch

HMMatch uses hidden Markov model to catch peptide fragmentation patterns (mass spectra peak intensity and variation). It provides a more accurate solution to the problem of recognizing seen peptide fragmentation patterns and thus a very good complement algorithm for current database search based peptide identification methods. As more and more mass spectra available to public, the next step work is to build relatively comprehensive HMM libraries for certain species (Human, yeast, etc.). Since it is fast to build single HMMs and using model extrapolation can help avoid building HMM from scratch for some peptides (with single nucleotide difference or post translational modification), mass spectra HMM libraries can be built in reasonable amount of time. Such libraries can be used as the first step of peptide identification by recognizing observed peptide mass spectra and thus improve peptide identification accuracy. Database search based methods can then be used to further identify the previously unobserved mass spectra patterns.

4.2.3 PepArML

PepArML uses random forest method to build model free framework to combine peptide identification results from multiple peptide identification softwares and provides better solution to separate true from false identifications. Current

PepArML implementation is based on three search engines: X!tandem, MASCOT and OMSSA and 20 metrics extracted from three search engines' output results. There is no feature design or selection involved in the whole process. So for the next step of the work,

- More novel features (retention time, etc.) can be designed and included in the PepArML framework. Poorly performing features can be removed.
- Different types of search engines (Sequest, Myrimatch, NIST MS-Search, HM-Match, etc.) can be included to provide a more comprehensive view of the peptide identification results.
- More stringent procedures may be designed to select true proteins for the unsupervised iterative procedure.
- Can refine statistical model to assess the significance of peptide identification results from PepArML.

After these refinements, PepArML can be used as a comprehensive framework to combine current peptide identification softwares.

Bioinformatics is a relatively new research area. Both high performance computing and machine learning techniques are widely used to solve challenge research problems in this field. This thesis presented two typical problems and three carefully designed solutions to demonstrate the power of both techniques for improving both speed and precision of bioinformatics applications and algorithms. These techniques can help biologists benefit from computer science research.

Bibliography

[AG96] S. Altschul and W. Gish. Local alignment statistics. *Methods in Enzymology*, 266:460–80, 1996.

[AGM+90] S. Altschul, W. Gish, E. Miller, E. Myers, and D. Lipman. A basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[ALPN03] D. C. Anderson, W. Li, D. G. Payan, and W. S. Noble. A new algorithm for the evaluation of shotgun peptide sequencing in proteomics: Support vector machine classification of peptide ms/ms spectra and sequest scores. *J. Proteome Res.*, 2(2):137–146, April 2003.

[Bar90] C. Bartels. Fast algorithm for peptide sequencing by mass spectrometry. *Biomedical and Environmental Mass Spectrometry*, 19:363–368, 1990.

[BBIK04] T. Baczek, A. Bucinski, A. R. Ivanov, and R. Kaliszan. Artificial neural network analysis for evaluation of peptide ms/ms spectra in proteomics. *Anal. Chem.*, 76(6):1726–1732, March 2004.

[BDH99] J. Burke, D. Davison, and W. Hide. d2_cluster: a validated method for clustering EST and full-length cDNA sequences. *Genome Res*, 9(11):1135–1142, Nov 1999.

[BE01] V. Bafna and N. J. Edwards. SCOPE: A probabilistic model for scoring tandem mass spectra against a peptide database. *Bioinformatics*, 17:13–21, 2001.

[BE03] Vineet Bafna and Nathan Edwards. On de novo interpretation of tandem mass spectra for peptide identification. In *Proceedings of the 7th annual international conference on computational molecular biology (RECOMB)*, 2003.

[BPC+99] R. C. Braun, K. T. Pedretti, T. L. Casavant, T. E. Scheetz, C. L. Birkett, et al. Three complementary approaches to parallelization of local BLAST service on workstation clusters. In *5th International Conference on Parallel Computing Technologies (PaCT)*, volume 1662. Lecture Notes in Computer Science (LNCS), 1999.

[CB04] R. Craig and R. C. Beavis. Tandem: matching proteins with tandem mass spectra. *Bioinformatics*, 20:1466–1467, 2004.

[CCFB06] R. Craig, J. C. Cortens, D. Fenyo, and R. C. Beavis. Using annotated peptide mass spectrum libraries for protein identification. *Journal of Proteome Research*, 5(8):1843–1849, 2006.

[CHV02] E. Coward, S. Haas, and M. Vingron. SpliceNest: visualization of gene structure and alternative splicing based on EST clusters. *Trends Genet*, 18(1):53–55, 2002.

[CKT+01] T. Chen, M. Y. Kao, M. Tepel, J. Rush, and G. M. Church. A dynamic programming approach to de novo peptide sequencing via tandem mass spectrometry. *J. Comput. Biol.*, 8(3):325–337, 2001.

[CN08] H. Choi and A. I. Nesvizhskii. Semisupervised model-based validation of peptide identifications in mass spectrometry-based proteomics. *J Proteome Res*, 7(1):254–265, January 2008.

[CRL+04] Ronghua Chen, Archie Russell, Guoya Li, Nicholas Tsinoremas, and Guy Cavet. Human transcript clustering. *Poster at RECOMB'04*, 2004.

[DAC+99] V. Dancik, T. Addona, K. Clauser, J. Vath, and P. A. Pevzner. De novo peptide sequencing via tandem mass spectrometry. *Journal of Computational Biology*, 6:327–342, 1999.

[DCF03] A. Darling, L. Carey, and W.-C. Feng. The design, implementation, and evaluation of mpiBLAST. In *ClusterWorld Conference & Expo and the 4th International Conference on Linux Clusters: The HPC Revolution 2003*, San Jose, CA, June 2003.

[DDK+06] F. Desiere, E. W. Deutsch, N. L. King, A. I. Nesvizhskii, P. Mallick, J. Eng, S. Chen, J. Eddes, S. N. Loevenich, and R. Aebersold. The PeptideAtlas project. *Nucleic Acids Res*, 34:D655–8, 2006.

[DL99] M. Deutsch and M. Long. Intron-exon structures of eukaryotic model organisms. *Nucleic Acids Research*, 27(15):3219–28, Aug 1999.

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

[EHFG05a] Joshua E Elias, Wilhelm Haas, Brendan K Faherty, and Steven P Gygi. Comparative evaluation of mass spectrometry platforms used in large-scale proteomics investigations. *Nature Methods*, 2:667–675, 2005.

[EHFG05b] Joshua E Elias, Wilhelm Haas, Brendan K Faherty, and Steven P Gygi. Comparative evaluation of mass spectrometry platforms used in large-scale proteomics investigations. *Nature Methods*, 2:667–675, 2005.

[FFM+05] L. Florea, V. Di Francesco, J. Miller, R. Turner, A. Yao, M. Harris, B. Walenz, C. Mobarry, G.V. Merkulov, R. Charlab, I. Dew, Z. Deng, S. Istrail, P. Li, and G. Sutton. Gene and alternative splicing annotation with AIR. *Genome Research*, 15(1):54–66, 2005.

[FHZ+98] L. Florea, G. Hartzell, Z. Zhang, G. M. Rubin, and W. Miller. A computer program for aligning a cdna sequence with a genomic dna sequence. *Genome Research*, 8(9):967–74, Sep 1998.

[FMSB+06] R. D. Finn, J. Mistry, B. Schuster-Bockler, S. Griffiths-Jones, V. Hollich, T. Lassmann, S. Moxon, M. Marshall, A. Khanna, R. Durbin, S. R. Eddy, E. L. L. Sonnhammer, and A. Bateman. Pfam: clans, web tools and services. *Nucleic Acids Res*, 34:D247–D251, 2006.

[FMW+06] Barbara E. Frewen, Gennifer E. Merrihew, Christine C. Wu, William Stafford Noble, and Michael J. MacCoss. Analysis of peptide MS/MS spectra from large-scale proteomics experiments using spectrum libraries. *Analytical Chemistry*, 78(16):5678–5684, 2006.

[FRR+05] Bernd Fischer, Volker Roth, Franz Roos, Jonas Grossmann, Sacha Baginsky, Peter Widmayer, Wilhelm Gruissem, and Joachim M Buhmann. Novohmm: A hidden markov model for de novo peptide sequencing. *Analytical Chemistry*, 77:7266–7273, November 2005.

[FVK+07] Jayson A. Falkner, Donna M. Veine, Maureen Kachman, Angela Walker, John R. Strahler, and Philip C. Andrews. Validated MALDI-TOF/TOF mass spectra for protein standards. *Journal of the American Society of Mass Spectrometry*, 18(5):850–855, 2007.

[FW] L. Florea and B. Walenz. in prep.

[GJS03] R. Giegerih, S. Jurtz, and J. Stoye. Efficient implementation of lazy suffix trees. *Software—Practice ad Experience*, 33:1035–1049, 2003.

[GMK+04] L. Y. Geer, S. P. Markey, J. A. Kowalak, L. Wagner, M. Xu, D. M. Maynard, X. Yang, W. Shi, and S. H. Bryant. Open mass spectrometry search algorithm. *Journal of Proteome Research*, 3:958–964, 2004.

[GMP96] M. Gelfand, A. Mironov, and P. Pevzner. Spliced alignment: A new approach to gene recognition. *Proc. Natl. Acad. Sci.*, 93:9061–9066, 1996.

[GMXL04] C. Grasso, B. Modrek, Y. Xing, and C. Lee. Genome-wide detection of alternative splicing in expressed sequences using partial order multiple sequence alignment graphs. *Pacific Symposium of Biocomputing*, 9:29–41, 2004.

[Gus77] D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology.* Cambridge University Press, 1977.

[HKB+07] R. E. Higgs, M. D. Knierman, A. Bonnerfreeman, L. M. Gelbert, S. T. Patil, and J. E. Hale. Estimating the statistical significance of peptide identifications from shotgun proteomics experiments. *J. Proteome Res.*, 6(5):1758–1767, May 2007.

[HM99] X. Huang and A. Madan. CAP3: A DNA sequence assembly program. *Genome Res*, 9:868–877, 1999.

[ISF+04] S. Istrail, G. G. Sutton, L. Florea, A. L. Halpern, C. M. Mobarry, R. Lippert, B. Walenz, H. Shatkay, I. Dew, J. R. Miller, M. J. Flanigan, N. J. Edwards, R. Bolanos, D. Fasulo, B. V. Halldorsson, S. Hannenhalli, R. Turner, S. Yooseph, F. Lu, D. R. Nusskern, B. C. Shue, X. H. Zheng, F. Zhong, A. L. Delcher, D. H. Huson, S. A. Kravitz, L. Mouchard, K. Reinert, K. A. Remington, A. G. Clark, M. S. Waterman, E. E. Eichler, M. D. Adams, M. W. Hunkapiller, E. W. Myers, and J. C. Venter. Whole genome shotgun assembly and comparison of human genome assemblies. *PNAS*, 101:1916–1921, 2004.

[JCL+06] J. R. Johnson, G. T. Cantin, B. Lu, J. D. Venable, D. Cociorva, and J. R. Yates. Reference library searching strategies in proteomics. Presentation at *54th ASMS Conference on Mass Spectrometry*, May 2006.

[JR85] B. H. Juang and L. R. Rabiner. A probabilistic distance measure between hmms. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 64(2):391–408, February 1985.

[JR90] B. H. Juang and L. R. Rabiner. The segmental k-means algorithm for estimating parameters of hidden markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38:1639–1641, 1990.

[KA93] S. Karlin and S. Altschul. Applications and statistics for multiple high-scoring segments in molecular sequences. *Proc Natl Acad Sci U S A*, 90(12):5873–7, Jun 1993.

[KAKB03] A. Kalyanaraman, S. Aluru, S. Kthari, and V. Brendel. Efficient clustering of large EST data sets on parallel computers. *Nucleic Acids Research*, 31(11):2963–2974, 2003.

[KBM+94] Anders Krogh, Michael Brown, I. Saira Mian, Kimmen Sjolander, and David Haussler. Hidden markov models in computational biology: applications to protein modeling. *Journal Molecular Biology*, 235(5):1501–1531, February 1994.

[KCW+07] Lukas Käll, Jesse D. Canterbury, Jason Weston, William S. Noble, and Michael J. Maccoss. Semi-supervised learning for peptide identification from shotgun proteomics datasets. *Nature Methods*, 4(11):923–925, October 2007.

[KDW+04] P. J. Kersey, J. Duarte, A. Williams, Y. Karavidopoulou, E. Birney, and R. Apweiler. The International Protein Index: An integrated database for proteomics experiments. *Proteomics*, 4(7):1985–1988, 2004.

[Ken02] W. Kent. Blat–the blast-like alignment tool. *Genome Research*, 12(4):656–64, Apr 2002.

[KNKA02] A. Keller, A. I. Nesvizhskii, E. Kolker, and R. Aebersold. Empirical statistical model to estimate the accuracy of peptide identifications made by ms/ms and database search. *Anal Chem*, 74(20):5383–5392, October 2002.

[LB04] Barbara Lin and Timothy Burcham. Using the human genome as a framework for sequence clustering and microarray design. *Poster at RECOMB'04*, 2004.

[LDE+06] Henry Lam, Eric Deutsch, James Eddes, Jimmy Eng, Nichole King, Sara Yang, Jeri Roth, Lisa Kilpatrick, Pedatsur Neta, Steve Stein, and Ruedi Aebersold. SpectraST: An open-source MS/MS spectra-matching library search tool for targeted proteomics. Poster at *54th ASMS Conference on Mass Spectrometry*, May 2006.

[LDE+07] Henry Lam, Eric W. Deutsch, James S. Eddes, Jimmy K. Eng, Nichole King, Stephen E. Stein, and Ruedi Aebersold. Development and validation of a spectral library searching method for peptide identification from MS/MS. *Proteomics*, 7(5):655–667, 2007.

[LE05] Jennifer Listgarten and Andrew Emili. Statistical and computational methods for comparative proteomic profiling using liquid chromatography-tandem mass spectrometry. *Molecular and Cellular Proteomics*, 17, 2005.

[LHP+00] F. Liang, I. Holt, G. Pertea, S. Karamycheva, S. Salzberg, and J Quackenbush. An optimized protocol for analysis of est sequences. *Nucleic Acids Research*, 28:3657–3665, 2000.

[LNRE05] Jennifer Listgarten, Radford M. Neal, Sam T. Roweis, and Andrew Emili. Multiple alignment of continuous time series. *Advances in Neural Information Processing Systems*, 4:419–434, 2005.

[MCJ03] K. Malde, E. Coward, and I. Jonassen. Fast sequence clustering using a suffix array algorithm. *Bioinformatics*, 19:1221–1226, 2003.

[MG77] A. M. Maxam and W. Gilbert. A new method for sequencing dna. *Proceedings of the National Academy of Science*, 74:560–564, 1977.

[Mot97] R. Mott. EST_GENOME: a program to align spliced DNA sequences to unspliced genomic DNA. *Computer Applications in the Biosciences*, 13(4):477–478, 1997.

[MSS05] W. Gary Mallard, O. David Sparkman, and Joan A. Sparkman. *NIST/EPA/NIH Mass Spectral Library (NIST 05) and NIST Mass Spectral Search Program (Version 2.0d)*. National Institute of Standards and Technology, June 2005. Available from `http://chemdata.nist.gov/mass-spc/Srch_v1.7/Ver20Man.pdf`.

[MW94] M. Mann and M. Wilm. Error-tolerant identification of peptides in sequence databases by peptide sequence tags. *Analytical Chemistry*, 66:4390–4399, 1994.

[MYL02] R. E. Moore, M. K. Young, and T. D. Lee. Qscore: an algorithm for evaluating sequest database search results. *J Am Soc Mass Spectrom*, 13(4):378–386, April 2002.

[MZL+03] B. Ma, K. Zhang, G. Lajoie, C. Doherty-Kirby, C. Hendrie, C. Liang, and M. Li. Peaks: Powerful software for peptide de novo sequencing by tandem mass spectrometry. *Rapid Communication in Mass Spectrometry*, 17(20):2337–2342, 2003.

[NCB] National Center for Biotechnology Information. *NCBI BLAST*. http://www.ncbi.nih.gov/BLAST/.

[NKKA03] A. I. Nesvizhskii, A. Keller, E. Kolker, and R. Aebersold. A statistical model for identifying proteins by tandem mass spectrometry. *Analytical Chemistry*, 75:4646–4658, 2003.

[OM02] J. Ogasawara and S. Morishita. Fast and sensitive algorithm for aligning ESTs to human genome. In *IEEE Computer Society Bioinformatics Conference (CSB'02)*, Stanford, CA, August 2002.

[OSA+05] G S Omenn, D J States, M Adamski, T W Blackwell, R Menon, H Hermjakob, R Apweiler, B B Haab, R J Simpson, J S Eddes, E A Kapp, R L Moritz, D W Chan, A J Rai, A Admon, R Aebersold, J Eng, W S Hancock, S A Hefta, H Meyer, Y K Paik, J S Yoo, P Ping, J Pounds, J Adkins, X Qian, R Wang, V Wasinger, C Y Wu, X Zhao, R Zeng, A Archakov, A Tsugita, I Beer, A Pandey, M Pisano, P Andrews, H Tammen, D W Speicher, and S M Hanash. Overview of the HUPO Plasma Proteome Project: Results from the pilot phase with 35 collaborating laboratories and multiple analytical groups, generating a core dataset of 3020 proteins and a publicly-available database. *Proteomics*, 5:3326–3245, August 2005.

[PGB02] J. Parkinson, D. Guiliano, and M. Blaxter. Making sense of EST sequences by CLOBBing them. *BMC Bioinformatics*, 3(31), October 2002.

[PHB93] D. J. Pappin, P. Hojrup, and A. J. Bleasby. Rapid identification of protein by peptide-mass fingerprinting. *Current Biology*, 3:327–332, 1993.

[PHL+03] G. Pertea, X. Huang, F. Liang, V. Antonescu, R. Sultana, S. Karamycheva, Y. Lee, J. White, F. Cheung, B. Parvizi, J. Tsai, and J Quackenbush. TIGR Gene Indices clustering tools (TGICL): a software system for fast clustering of large EST datasets. *Bioinformatics*, 19(5):651–652, 2003.

[PPCC99] D. N. Perkins, D. J. Pappin, D. M. Creasy, and J. S. Cottrell. Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, 20:3551–3567, 1999.

[PSL+04] T. Pohar, H. Sun, S. Liyanarachchi, S. James, S. Stapleton, and R. Davuluri. A bioinformatics approach toward identification of genes involved in hematopoiesis and leukemia. *Poster at RECOMB'04*, 2004.

[Rab89] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

[RMAM+04] K. A. Resing, K. Meyer-Arendt, A. M. Mendoza, L. D. Aveline-Wolf, K. R. Jonscher, K. G. Pierce, W. M. Old, H. T. Cheung, S. Russell, J. L. Wattawa, G. R. Goehle, R. D. Knight, and N. G. Ahn. Improving reproducibility and sensitivity in identifying human proteins by shotgun proteomics. *Anal. Chem.*, 76(13):3556–3568, July 2004.

[SCH+82] F. Sanger, A. R. Coulson, G. F. Hong, D. F. Hill, and G. B. Petersen. Nucleotide sequence of bacteriophage λ dna. *Journal of Molecular Biology*, 162:729–773, 1982.

[SCK04] M. K. Sakharkar, V. Chow, and P. Kangueane. Distributions of exons and introns in the human genome. *In Silico Biology*, 4:32, 2004.

[SKG04] Alexander Sczyrba, Jan Krüger, and Robert Giegerich. e2g - a web-based tool for efficiently aligning genomic sequence to EST and cDNA data. *Poster at RECOMB'04*, 2004.

[SKN+06] Steve Stein, Lisa Kilpatrick, Pedatsur Neta, Jeri Roth, and Xiaoyu Yang. A reference library of peptide ion fragmentation spectra. Poster at *54th ASMS Conference on Mass Spectrometry*, May 2006.

[SKSS03] F. Schutz, E. A. Kapp, R. J. Simpson, and T. P. Speed. Deriving statistical models for predicting peptide tandem MS product ion intensities. *Biochem. Soc. Trans.*, 31:1479–1483, 2003.

[SNC77] F. Sanger, S. Nickolen, and A. R. Coulson. Dna sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Science*, 74:5463–5467, 1977.

[SS94] S. Stein and D. Scott. Optimization and testing of mass spectral library search algorithms for compound identification. *Journal of the American Society of Mass Spectrometry*, 5:859–866, 1994.

[Ste94] S. Stein. Estimating probabilities of correct identification from results of mass spectral library searches. *Journal of the American Society of Mass Spectrometry*, 5:316–323, 1994.

[Ste95] S. Stein. Chemical substructure identification by mass spectral library searching. *Journal of the American Society of Mass Spectrometry*, 6:644–655, 1995.

[STN08] Brian C. Searle, Mark Turner, and Alexey I. Nesvizhskii. Improving sensitivity by probabilistically combining results from multiple ms/ms search methodologies. *J. Proteome Res.*, 7(1):245–253, January 2008.

[TJ97] J. A. Taylor and R. S. Johnson. Sequence database searches via de novo peptide sequencing by tandem mass spectrometry. *Rapid Commun Mass Spectrom*, 11(9):1067–1075, 1997.

[TSF+05] Stephen Tanner, Hongjun Shu, Ari Frank, Ling-Chi Wang, Ebrahim Zandi, Marc Mumby, Pavel A. Pevzner, and Vineet Bafna. Inspect: Fast and accurate identification of post-translationally modified peptides from tandem mass spectra. *Analytical Chemistry*, 77:4626–4639, 2005.

[TSY03] D.L. Tabb, A. Saraf, and J.R. Yates. GutenTag: high-throughput sequence tagging via an empirically derived fragmentation model. *Analytical Chemistry*, 75(23):6415–6421, December 2003.

[UZQA06] P. J. Ulintz, J. Zhu, Z. S. Qin, and P. C. Andrews. Improved classification of mass spectrometry database search results using newer machine learning approaches. *Mol Cell Proteomics*, 5(3):497–509, March 2006.

[VHS03] Natalia Volfovsky, Brian Haas, and Steven Salzberg. Computational discovery of internal micro-exons. *Genome Research*, 13:1216–1221, 2003.

[WBB+03] D. Wheeler, T. Barrett, D. A. Benson, S. H. Bryant, K. Canese, et al. Database resources of the national center for biotechnology. *Nucleic Acids Research*, 31:28–33, 2003.

[WCO01] S. Wheelan, D. Church, and J. Ostell. Spidey: a tool for mrna-to-genomic alignments. *Genome Research*, 11(11):1952–7, Nov 2001.

[WF05] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann, San Francisco, 2005.

[WLM+05] W.E. Wolski, M. Lalowski, P. Martus, R. Herwig, P. Giavalisco, J. Gobom, A. Sickmann, H. Lehrach, and K. Reinert. Transformation and other factors of the peptide mass spectrometry pairwise peak-list comparison process. *BMC Bioinformatics*, 6(1):285, 2005.

[WLT05] X. Wu, W.-J. Lee, and C.-W. Tseng. Estmapper: Efficiently aligning dna sequences to genomes. In *Fourth IEEE International Workshop on High Performance Computational Biology (HiCOMB 2005)*, 2005.

[WOOT03] H. Wang, T. Ong, B. Ooi, and K. Tan. BLAST++: A Tool for BLASTing Queries in Batches. In *Proceedings of the 1st Asia-Pacific Bioinformatics Conference*, Adelaide, Australia, February 2003.

[WUB] Washington University School of Medicine. *WU BLAST*. http://blast.wustl.edu/blast/README.html.

[WW05] Thomas Wu and Colin Watanabe. GMAP: a genomic mapping and alignment program for mRNA and EST sequences. *Bioinformatics*, 21(9):1859–1875, May 2005.

[WYC06] Yunhu Wan, Austin Yang, and Ting Chen. PepHMM: A hidden markov model based scoring function for tandem mass spectrometry. *Analytical Chemistry*, 78(2):432–437, January 2006.

[YECB96] J. R. Yates, J. K. Eng, K. R. Clauser, and A. L. Burlingame. Search of sequence databases with uninterpreted high-energy collision-induced dissociation spectra of peptides. *Journal for the American Society of Mass Spectrometry*, 7:1089–1098, 1996.

[YMG+98] J. R. Yates, S. F. Morgan, C. L. Gatlin, P. R. Griffin, and J. K. Eng. Method to compare collision-induced dissociation spectra of peptides: potential for library searching and subtractive analysis of peptide tandem mass spectra. *Analytical Chemistry*, 70(17):3557–65, September 1998.

[Zha04] Z. Zhang. Prediction of low-energy collision-induced dissociation spectra of peptides. *Analytical Chemistry*, 76(14):3908–3922, 2004.

[ZSWM00] Z. Zhang, S. Schwartz, L. Wagner, and W. Miller. A greedy algorithm for aligning DNA sequences. *Journal of Computational Biology*, 7:203–214, 2000.