Université de Montréal

Deep Learning Algorithms for Database-Driven Peptide Search

par

Jeremie Zumer

Departement d'Informatique et de Recherche Operationelle
Faculté des arts et des sciences

Thèse présentée en vue de l'obtention du grade de
Philosophiæ Doctor (Ph.D.)
en Informatique

13 Sept 2023

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée

Deep Learning Algorithms for Database-Driven Peptide Search

présentée par

Jeremie Zumer

a été évaluée par un jury composé des personnes suivantes :

François Major
(président-rapporteur)

Sébastien Lemieux
(directeur de recherche)

Paul François
(membre du jury)

Mathieu Lavallée-Adam
(examinateur externe)

Pierre Thibault
(représentant du doyen de la FESP)

Résumé

La protéomique moderne – l'analyse à grande échelle des protéines (Graves and Haystead, 2002) - dépend fortement de l'analyse de données expérimentales de série chronologique complexes. Dans un flux de travail typique de spectrométrie de masse en *shotgun*, où l'objectif est d'identifier les protéines en solution, un mélange complexe de protéines est préparé, digéré, fractionné par exemple par catégorie de masse ou par hydrophobicité, ionisé et injecté dans un spectromètre de masse, ce qui donne ce que l'on appel un spectre de masse. Dans le mode de spectrométrie de masse en tandem, il représente des signaux à la résolution des acides aminés sur les peptides présentes. Le spectre doit être nettoyé pour se prêter à une analyse plus approfondie, puis les pics définis par les couples de valeurs m/z et d'intensité dans le spectre peuvent être mis en correspondance avec une séquence de pics attendue selon la séquence hypothétique du peptide présent dans le spectre (qui sont souvent obtenus par digestions in-silico du protéome de l'espèce source) ; il s'agit du processus d'identification des peptides proprement dit.

Dans ce travail, nous sélectionnons et résolvons certaines limitations actuelles spécifiques au côté informatique de la recherche sur l'identification des peptides. Nous introduisons d'abord un nouveau moteur d'identification axé sur la recherche. Une question majeure à la frontière actuelle en protéomique est l'intégration et la viabilité de nouveaux algorithmes basés sur l'apprentissage profond dans un contexte d'identification. Très peu de travail a été effectué sur ce sujet jusqu'à présent, Prosit (Gessulat et al., 2019) étant le seul logiciel de ce type à voir l'intégration dans un moteur de recherche préexistant, au meilleures de nos connaissances (bien que des algorithmes de *rescoring* comme Percolator (Käll et al., 2007) , qui utilisent généralement des algorithmes d'apprentissage automatique plus classiques, sont habituellement utilisés depuis un certain temps maintenant, ils sont simplement appliqués comme étape de post-traitement et non intégrés dans le moteur). Pour étudier ce problème, nous développons et présentons un nouvel algorithme d'apprentissage en profondeur qui effectue la prédiction de la longueur des peptides à partir d'un spectre (le premier algorithme de ce type), et calculons des métriques basées sur cette prédiction. Nous utilisons l'algorithme résultant pour démontrer des identifications de peptides constamment améliorées après intégration dans notre engin. De plus, nous proposons un nouvel algorithme de prédiction de

spectres complets (conforme à PredFull (Liu et al., 2020) plutôt qu'à Prosit) ainsi qu'un nouvel algorithme et paradigme de rescoring basé sur la forêt aléatoire, que nous intégrons encore à notre moteur de recherche. En somme, les outils d'apprentissage en profondeur que nous proposons démontrent une amélioration de plus de 20% des taux d'identification de peptides à un seuil de taux de fausse découverte (FDR) de 1%. Ces résultats suggèrent pour la première fois que les algorithmes d'apprentissage profonds proposés en protéomique peuvent en effet largement améliorer les identifications.
Mots clefs : Apprentissage profond, Apprentissage automatique, peptide, protéomique

Abstract

Modern proteomics – the large-scale analysis of proteins (Graves and Haystead, 2002) – relies heavily on the analysis of complex raw experimental, time series-like data. In a typical shotgun mass spectrometry workflow where the goal is to identify proteins in solution, a complex protein mixture is prepared, digested, fractionated for example by mass range, ionized and injected into a mass spectrometer, resulting in a so-called mass spectrum which, in tandem mass spectrometry, achieves obtain amino acid-resolution signals for the detected peptides. The spectrum must be cleaned up to become suitable for further analysis, then the peaks defined by the m/z to intensity values in the spectrum can be matched to some expected peak sequence from a set of candidate peptides (which are often simply in silico digests from the source specie's proteome), which is the process of peptide identification proper.

In this work, we select and solve some current limitations in the computational side of peptide identification research. We first introduce a new, research-oriented search engine. A major question at the boundary of current proteomics research is the integration and viability of new deep learning-driven algorithms for identification. Very little work has been done on this topic so far, with Prosit (Gessulat et al., 2019) being the only such software to see integration in an existing search engine, as far as we are aware (although rescoring algorithms like Percolator (Käll et al., 2007), which typically use more classical machine learning algorithms, have been in routine use for a while by now, they are merely applied as a postprocessing step and not integrated in the engine per se). To investigate this, we develop and present a new deep learning algorithm that performs peptide length prediction from a spectrum (a first, as far as we are aware). We compute metrics based on this prediction that we use during rescoring, and demonstrate consistently improved peptide identifications. Moreover, we propose a new full spectrum prediction algorithm (in line with PredFull (Liu et al., 2020) rather than Prosit) and a novel, random forest-based rescoring algorithm and paradigm, which we integrate within our search engine. Altogether, the deep learning tools we propose show an increase of over 20% in peptide identification rates at a 1% false discovery rate (FDR) threshold. These results provide strong evidence that deep learning-based tools proposed for proteomics can greatly improve peptide identifications.

Keywords: deep learning, machine learning, peptide, proteomics

Contents

List of Tables

List of Figures

Symbols and Abbreviations

2D-PAGE	Two dimensional polyacrylamide gel electrophoresis
SDS-PAGE	Sodium Dodecyl Sulfate polyacrylamide gel electrophoresis
MSE	Mean Square Error
TDA	Target-Decoy Approach
FDR	False Discovery Rate
ML	Machine Learning
DL	Deep Learning
SVM	Support Vector Machine
MAP	MHC-I Associated Peptide
MHC-I	Major Histocompatiblity Complex I

ESI	Electro-Spray Ionisation
MALDI	Matrix Assisted Laser Desorption/Ionization
TOF	Time Of Flight
HCD	Higher-energy Collision Dissociation ("higher energy" refers to the voltage of the C-trap, not collision energy)
ETD	Electron Transfer Dissociation
CID	Collision-Induced Dissociation
MLP	Multi-Layer Perceptron
KL(D)	Kullback-Leibler (Divergence)
NN	Neural Network
OLS	Ordinary Least Squares
CNN	Convolutional Neural Network
FNN	Fully-connected Neural Network

LSTM	Long Short-Term Memory
BiLSTM	Bidirectional Long Short-Term Memory
RNN	Recurrent Neural Network (not to be confused with "Recursive Neural Network", which may involve connections from a neuron to itself)
AE	Auto-Encoder
VAE	Variational Auto-Encoder
HPLC	High-Performance (formerly High-Pressure) Liquid Chromatography
MS	Mass Spectrum (or Mass Spectra), or Mass Spectrometry
PSM	Peptide-Spectrum Match
PIN	Percolator INput file format
POUT	Percolator OUTput file format
FDP	False-Discovery Proportion (sometimes "Factual FDR" as it is based on ground truth, not estimation e.g. with decoys)

PTM	Post-Translational Modification
RT	Retention Time
iRT	indexed Retention Time
Da	Dalton
PPM	Parts Per Million
Th	Thompson
KB	Knowledge Base
CDF	Cumulative Distribution Function
P-value	Probability value
MGF	Mascot Generic Format
NCE	Normalized Collision Energy
RNHS	ReNormalized HyperScore (X!Tandem HyperScore with normalization by total intensity instead of maximum intensity)

Acknowledgement

I would like to thank the Lemieux Lab and IRIC for their continuous support during this journey. I am especially grateful to Assya Trofimov, Guillaume Poirier-Morency and Sebastien Lemieux for interesting and fruitful discussions over the years.

Chapter 1

Introduction

This work presents a software-oriented approach to proteomics research. Its primary goal is to improve peptide identifications from mass spectrometry data after the wetlab experiments are already complete, in part by developing deep learning algorithms using the large amount of data that has been made available relatively recently in the field. This represents a multi-disciplinary endeavor at the cutting edge of proteomics and machine learning research. Here we attempt to cover the most important background elements of those disparate domains, and discuss some of the important challenges arising from the qualities of the data and tools relevant to this work.

1. Proteomics

Proteomics is the study of proteomes and their function (Dupree et al., 2020). Many tasks and paradigms are considered within the scope of proteomics (Han et al., 2008a), such as quantification (Urban, 2016) or identification (Bandeira, 2010). Proteomics protocols can also entail many strategies, such as bottom-up (in the sense of analysing a collection of peptides, also called peptide-centric), top-down (refering to analysing protein mixtures rather than first breaking them down into peptide mixtures) (Reid and McLuckey, 2002) or middle-down (using limited digestion of proteins) (Duncan et al., 2010; Han et al., 2008a), and can also be differentiated between sequence-based and structure-based methods (Serpa et al., 2012; Krissinel, 2007; Shin et al., 2008). Another common dichotomy in sequential proteomics is the Data-Dependent Acquisition (DDA) vs Data-Independent Acquisition (DIA) methologoies (Li et al., 2021; Kitata et al., 2022). This non-exhaustive list demonstrates the wide breadth of methods and techniques in use in proteomics, which displays the versatility of proteomics for various types of workflow useful to different analyses of interest.

In this work, we focus specifically on the bottom-up identification setting with data from data-dependent acquisition experiments. Moreover, our results do not consider structural

data due to various limitations on availability, quality, and resource requirements, which represent various bottlenecks for deep learning-based methods.

In our study setting, mass spectrometry has shown to be both fast and accurate, as well as being capable of generating large amounts of data (Han et al., 2008a,b). Indeed, some large, high-quality datasets of proteomics data acquired by mass spectrometry are readily available to the public, such as ProteomeTools (Zolg et al., 2017), the NIST mass spectra library (Stein, 2008), or the MassIVE-KB invivo and synthetic peptide datasets (Wang et al., 2018) for human data. Unfortunately, such quality data is less available for other organisms. As a result, the work presented in this thesis is primarily evaluated against those high-quality human-centric datasets. We occasionally use other datasets to demonstrate generalization and limitations across species or modalities, notably the One Hour Yeast Proteome (Hebert et al., 2014), for instance.

This chapter is organized in three main segments: in the first, we describe the organization of mass spectrometry experiments as per our study setting; that segment is further organized into wetlab and drylab parts, mirroring the usual workflow in these experiments. In the second, we review deep learning bases, including some prerequisite notions from machine learning and statistical learning (i.e. pre-deep learning, so to speak) as well as the common deep learning model building blocks. In the third, we cover some recent and important developments at the intersection of deep learning and mass spectrometry-based proteomics relevant to the identification setting this work covers.

1.1. Mass Spectrometry

Mass spectrometry is the most suitable tool for the mass processing of proteins at the scale needed for omics experiments. Here we summarize key concepts in mass spectrometry, especially in the context of bottom up, peptide identification proteomics using DDA.

1.1.1. Spectra, Peptides. Before describing the workflow of an identification experiment using a mass spectrometry, it is useful to understand the fragmentation process within a mass spectrometer as well as the data involved in the process. Peptides are small amino acid chains. Longer peptides are called polypeptides, and polypeptides with molecular mass of 10 kilo-dalton (kDa) or more are called proteins. Peptide sequences are commonly presented by strings of 1 or 3-letter codes for their amino acids in a linear sequences. Table 1 lists common amino acids and their corresponding codes. The goal in identification-based proteomics is to identify peptides present in a prepared, purified mixture, usually to resolve the proteins that were originally present in the sample of interest.

After a peptide sample is injected into a mass spectrometer, it outputs a so-called mass spectrum. Mass spectra are intensity signals presented as a function of mass to charge (m/z, whose unit is commonly referred to as the thompson (Th), although neither thompson nor

Amino Acid	3 Letters	1 Letter	Mass (Da)
Alanine	Ala	A	71.0788
Arginine	Arg	R	156.1875
Aspartic Acid	Asp	D	114.1038
Asparagine	Asn	N	115.0886
Cysteine	Cys	C	103.1388
Glutamic Acid	Glu	E	129.1155
Glutamine	Gln	Q	128.1307
Glycine	Gly	G	57.0519
Histidine	His	H	137.1411
Isoleucine	Ile	I	113.1594
Leucine	Leu	L	113.1594
Lysine	Lys	K	128.1741
Methionine	Met	M	131.1926
Phenylalanine	Phe	F	147.1766
Proline	Pro	P	97.1167
Serine	Ser	S	87.0782
Threonine	Thr	T	101.1051
Tryptophane	Trp	W	186.2132
Tyrosine	Tyr	Y	163.1760
Valine	Val	V	99.1326

Table 1 – List of common amino acids, their 3- and 1-letter codes, and their average molecular mass

dalton are standard international units, and some authors prefer referring to mass to charge units in daltons (Da) or "atomic mass units" (AMU or U)). An example raw spectrum is presented in Figure 2A. In Figure 2B, the same spectrum has been processed through typical processing phases, namely smoothing (smoothing out variances in intensity measurement that are likely due mostly to instrument sensitivity, i.e. experimental noise), baseline correction (to remove the base intensity bias, another source of experimental noise due to calibration, which shifts over time during a mass spectrometry experiment), and intensity normalization (to put the intensities within a value regime more easily processed by further tools) (Stanford et al., 2016). In Figure 2C, m/z-intensity levels that correspond to peaks are identified. This final processed spectrum is a graphical representation of the signal that would then be analyzed by peptide identification software. Figure 2E shows this final processed spectrum with the theoretical fragment masses generated in-silico (Figure 2D) matching the m/z values in the experimental spectrum within 20 ppm (parts per million, calculated as in Equation 1.1). As shown in the figure, few of the theoretical peaks actually match in the experimental spectrum, indicating that many expected peaks are missing.

$$\frac{|\mathtt{mz}_{\mathtt{theoretical}} - \mathtt{mz}_{\mathtt{experimental}}|}{\mathtt{mz}_{\mathtt{experimental}}} 10^6 \tag{1.1}$$

While different fragmentation methods result in different fragmentation patterns, Figure 1 shows an example fragmentation diagram on a given peptide. The labels (e.g. b and y and the following numbers) are composed of two parts: an ion series identifier (the letter), and a number representing the sequence number of the ion in the series. Series labeled from letters at the end of the alphabet, such as x, y or z, represent fragmentation starting at the N terminal, while those labeled from the start of the alphabet, like a, b or c, represent fragmentation beginning at the C terminal (Wysocki et al., 2005; CHONG and LEONG, 2012).

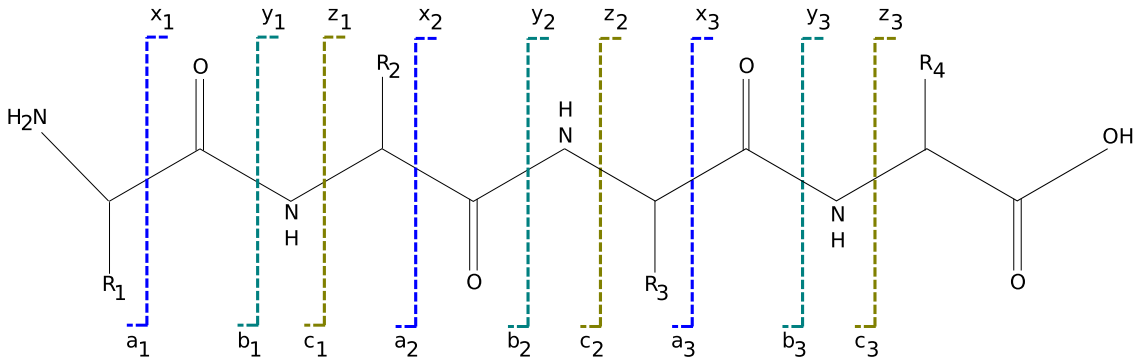


Figure 1 – Diagram representation of the fragmentation process of a peptide injected in a mass spectrometer. R_n is the Nth residue (i.e. what is identified in the amino acid string representation of a peptide). The peptide retains the same overall structure during fragmentation as residues fragments off a side of the molecule.

Amino acid masses cannot easily be resolved to m/z peaks in the spectrum even beyond noise in the data so easily due to issues with mass conflicts: some di- and tri-amino acids have the same masses, as do some amino acids and di-amino acids. Indeed, some amino acids are indistinguishable from their mass alone, especially when taking into account the resolution of some instruments. In Table 1, Lysine and Glutamine have nearly identical mass, while Leucine and Isoleucine, which have the same molecular composition but not the same structure, have exactly identical masses. Other examples include the di-amino acids VA vs LG and AD vs SV, the di-amino acid GE vs the single amino acid W, or GV vs R, and more.

Beyond the noisy nature of the data, mass spectra are complicated to analyse with our current understanding of proteomics. As the annotations in Figure 2D suggest, many of the peaks cannot be properly explained (that is, either they come from contaminants (Brewis and Brennan, 2010), are artifacts of the processing pipeline, or are simply not understood at all, and further work in the area are required to complete our understanding of them (Vu et al., 2014)). Some are known to derive from unexpected physical effects like collision with water molecules in the spectrometer (Neta et al., 2014), and many peaks are missing unexpectedly (Cleveland and Rose, 2013). This has long affected so-called de novo sequencing

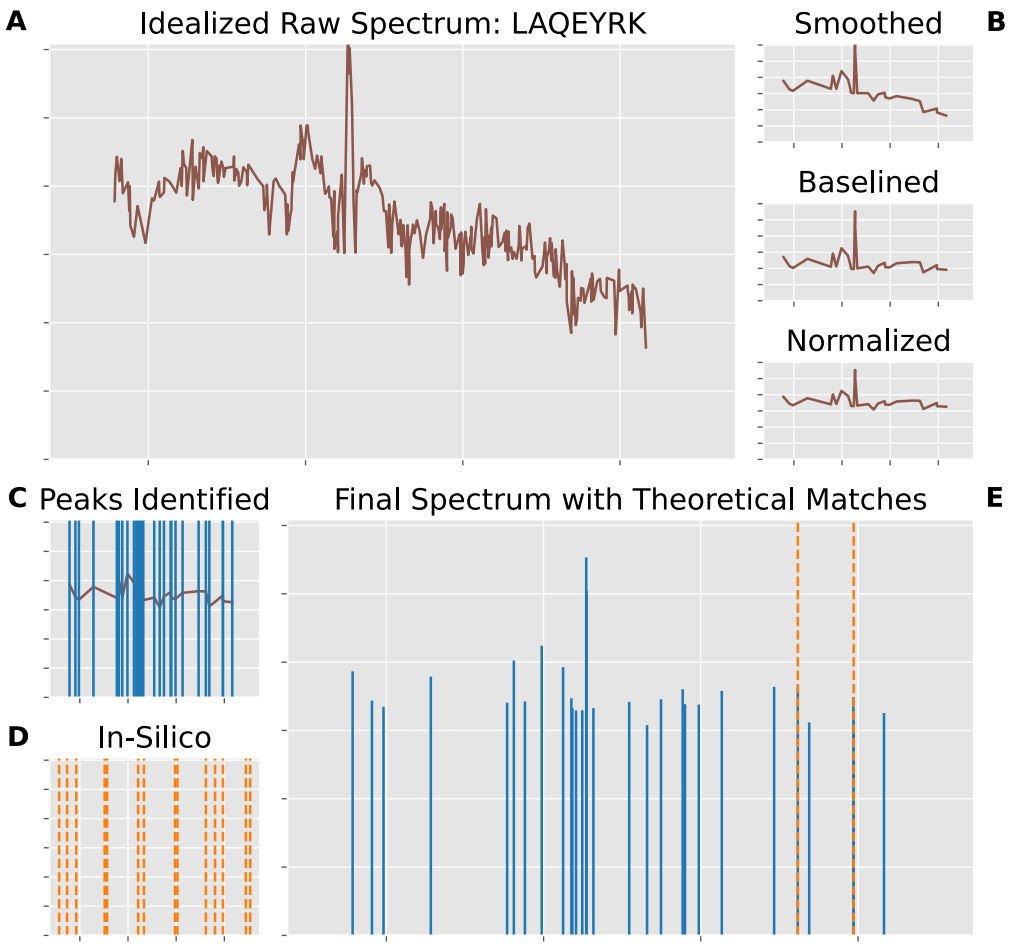


Figure 2 – Idealized mass spectrum with corresponding peptide in various stages of a usual preprocessing pipeline. **A**: Raw. **B**: From top to bottom: after smoothing, baseline correction ("baselining") and intensity normalization. **C**: With peaks identified. **D**: The corresponding in-silico generated theoretical spectrum for the peptide sequence. **E**: Final peaks with overlaid matching theoretical peaks.

software, which attempts to determine the peptide sequence in a spectrum based solely on the mass spectrometer output (e.g. the spectrum, precursor peaks, retention time, and other metadata; as opposed to methods that rely on spectral databases, which are the more common and successful software at present) (Cleveland, 2013; Fischer et al., 2005; Lu and Chen, 2003), and negatively impacts the performance of other methodologies as well. Moreover, peptide separation (explained in the below sections) may be incomplete and cause the mass spectra to show the signal associated with two peptides instead of just one (referred to as chimeric spectra) (Houel et al., 2010). For these reasons, peptide identification requires sophisticated algorithms and complicated processing for confident results.

1.1.2. Workflow. In shotgun mass spectrometry experiments, a typical workflow can be decomposed into 6 major phases (Graves and Haystead, 2002; Brewis and Brennan, 2010):

(1) Protein mixture isolation;

(2) Protein separation, e.g. by gel electrophoresis;

(3) Cleavage, e.g. by digestion with Trypsin;

(4) Peptide separation, e.g. by liquid chromatography;

(5) Mass spectrometric measurement;

(6) Software analysis.

This workflow is also illustrated in Figure 3. In modern experiments, the second step is often skipped (Brewis and Brennan, 2010), i.e. all proteins may be digested (step 3) and then separated (step 4) without a prior fractionation step at the protein level.

A vast array of technology combinations can be used to satisfy the steps of this workflow at every stage. Due to the wide variety of options at each stage, and because this work primarily focuses on computational approaches for peptide identifications, we only cover a few possibilities to illustrate their impact on the overall identification process in the following text.

1.1.3. Sample Preparation. Samples are typically selected from cell mixtures or biological fluids (Brewis and Brennan, 2010) in most real-world conditions, however various experimental settings may call for other preparations, such as synthetic peptide mixtures (Frank, 2002; Zolg et al., 2017) or immunopeptides (Laumont et al., 2018, 2016), for instance. In the more usual case, the initial cell sample may be subjected to subcellular fractionation, just as a biological fluid sample may be subjected to protein depletion (Brewis and Brennan, 2010), or the entire sample may be used, depending on the goals of the experiments. In any case, the resulting proteins (if applicable, e.g. unlike in the case where the input product is already a set of peptides as in the above examples) are solubilized in preparation for further processing. Unlike in other omics fields like genomics and transcriptomics, physiochemical properties in the sample differ. Therefore, no universal buffer or method for sample preparation exists as of yet (Dupree et al., 2020). Meanwhile, just as any steps in the usual workflow has downstream consequences throughout the analysis process, and just as inefficiencies at previous steps tend to compound errors downstream, the sample preparation quality can be considered most impactful in the success of proteomics experiments (Dupree et al., 2020).

1.1.4. Protein Separation. Protein separation, or fractionation, may optionally be performed before peptide digestion (Dupree et al., 2020; Brewis and Brennan, 2010). Good fractionation strategies are important to ensure the qualities of the sample are sufficiently uniform for further processing (Brewis and Brennan, 2010). In some experiments, the entire sample is digested and no protein separation is performed (Brewis and Brennan, 2010). If protein separation is to be performed, various technologies can be used, although perhaps the most common is two dimensional polyacrylamide gel electrophoresis (2D-PAGE) (Dupree

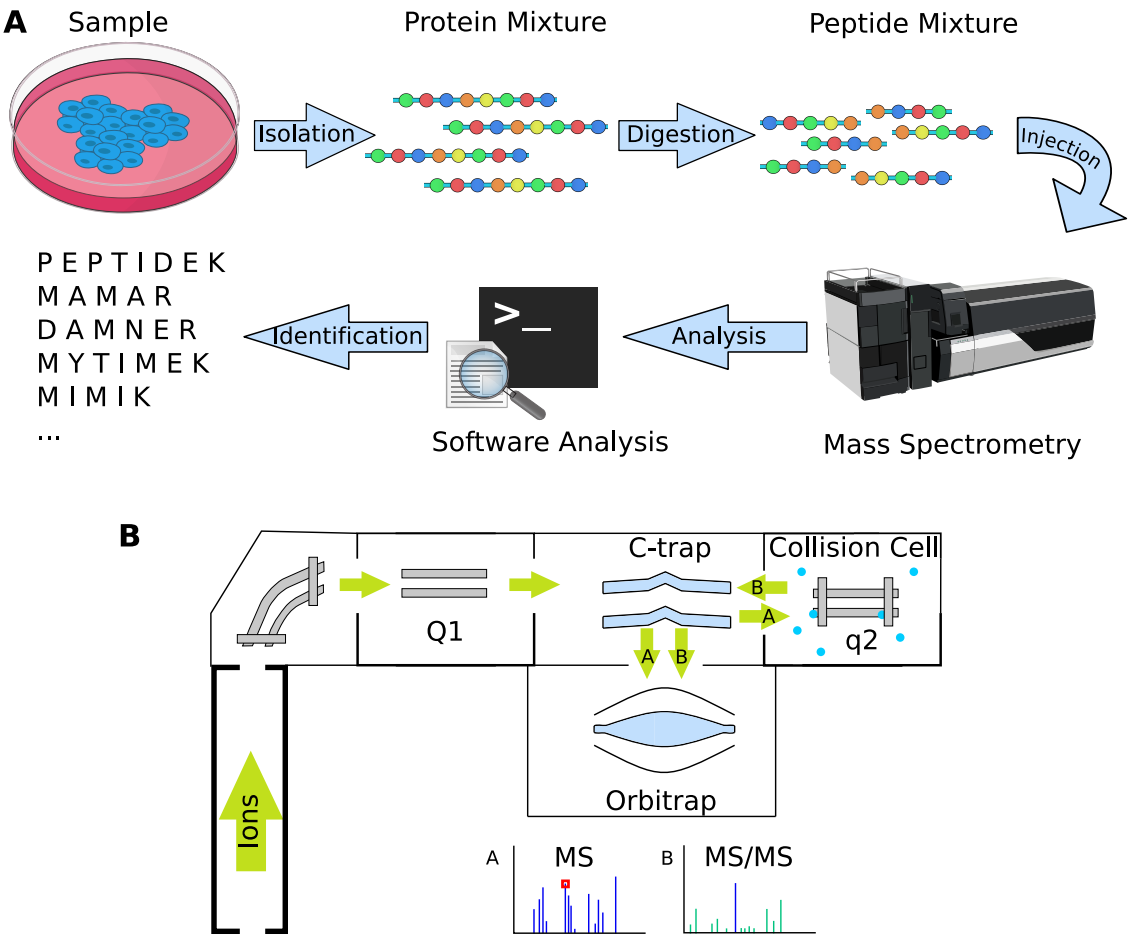


Figure 3 – Illustration of a typical shotgun mass spectrometry workflow. **A**: The overall workflow steps, from sample preparation to computational peptide identification. **B** Simplified diagram of a mass spectrometer (a QExactive). Samples enter the device and are ionised (e.g. by ESI). The ions are then focused through a lens and filtered for charge in the bended guide (non-charged peptides are not guided and thus do not proceed into the next chamber). They enter Q1, a quadrupole that operates in mass filtering mode. Species that survive the mass to charge filtering are then guided to the orbitrap mass analyzer (along path A), a device which traps ions in orbit around the central core, allowing measuring the mass spectrum. In tandem mass spectrometry, the ions also traverse the C-trap (a radio frequency-based trap used to cool down the ions before the next phase) to a second quadrupole, q2, in a collision chamber containing neutral gas. The resulting fragments are once again submitted to the orbitrap as they come back along path B from the HCD round. An MS/MS spectra is collected based on the image current caused by the current induced on the outer electrode by the movement of ions in the trap.

et al., 2020; Graves and Haystead, 2002; Brewis and Brennan, 2010). Other methods, like Sodium Dodecyl Sulfate (SDS) PAGE, which performs one dimensional separation, as opposed to 2D-PAGE, may be used alternatively (Graves and Haystead, 2002). SDS-PAGE has high resolution range and is simple to perform, but it can only separate proteins along one

Enzyme	Cleavage	Note
Trypsin	K or R not followed by P	Most common choice
Chymotrypsin	W, F or Y; L, M, A, D or E	More efficient for W, F, Y
Elastase	N, V, L, I, G or S	
LysC	K	Longer peptides than trypsin
LysN	Any followed by K	Longer peptides, more highly charged in ETD
GluC	E, some D	
ArgC	R	
AspN	Any followed by D	

Table 2 – List of some common enzymes used for protein cleavage. Cleavage happens after the listed amino acids, as per the conditions described. Table adapted from Zhang et al. (2013). ETD: electron-transfer dissociation

property (i.e. mass difference). By comparison, 2D methods like 2D-PAGE can separate proteins along two dimensions (as the name implies), such as both mass and net charge, which is often sufficient to identify post-translationally modified proteins, for example (Graves and Haystead, 2002).

1.1.5. Digestion. While the enzyme Trypsin is a popular choice for protein digestion due to its high efficiency, predictability, and generated peptide sizes (Dupree et al., 2020), many other choices (some listed in Table 2) may be viable depending on experimental context. In practice, around 96% of data in the Global Proteome Machine Database relate to trypsic peptides as of 2020 (Dupree et al., 2020). This may be problematic for non-standard workflows, such as proteomics experiments aiming to identify cell surface markers (Laumont et al., 2016), as the relative lack of data can make successful analysis workflow development more complicated. In addition, computational tools are tested on available data, which may induce a bias for trypsic peptide identifications (this can potentially lead to high-confidence, low quality (in fact, false) hits in peptide identification engines, for example if false discovery rate estimation is overtuned toward trypsic peptide patterns).

1.1.6. Peptide Separation. Similarly to protein separation, many technologies can be used, such as capillary electrophoresis or high-performance liquid chromatography (HPLC) (Dupree et al., 2020; Graves and Haystead, 2002; Zhang et al., 2013) (the latter having largely supplanted the former in common experimental settings (Dupree et al., 2020)), as well as many other approaches, such as cation-exchange chromatography or multidimensional peptide identification (Graves and Haystead, 2002) may also be used. In addition, there are many ways to operate the same basic technology: reversed-phase HPLC and size-exclusion chromatography are two popular operation mode for HPLC-based workflows (Dupree et al.,

2020). These different technologies can leave different patterns in the mass spectrum generated by a mass spectrometry experiment, as they may be more or less efficacious in purifying peptide mixtures, depending on the overlap in properties in the sample (for example, reversed-phased HPLC separates peptides based on hydrophobicity while size-exclusion chromatography separates by peptide size). Poor separation can result in "chimeric spectra" (Houel et al., 2010), where the tandem mass spectrum captured by the instrument actually contains peaks at m/z values corresponding to multiple peptides, affecting downstream analysis. This can also potentially lead to batch effects if one were interested in combining multiple datasets for tool development.

There are many ways to analyse the fractionated peptide mixture by mass spectrometry. The general steps are: ionisation, mass analysis, and fragmentation. The fragmentation step can be "repeated" zero or more times, If it is not applied, the experiment is sometimes referred to as "MS1", or simply "MS", and provides peptide-level m/z and intensity sensitivity. Identification typically uses the peptide mass fingerprinting paradigm in this case (Zhang et al., 2013). If it is performed once, the experiment may be referred to as "MS2", "MS/MS" or "tandem" mass spectrometry (Graves and Haystead, 2002; Zhang et al., 2013), which can provide amino acid-level resolution. In any case, the mass spectrometer detects a signal corresponding to a fragment's mass to charge, and this record forms the mass spectrum that is further analyzed using computational tools.

Multiple different technologies can be used for ionisation and injection. The most common choice today is electrospray ionisation (ESI). Other possibilities include Matrix-Assisted Laser Desorption/Ionisation (MALDI). Similarly, intensity measurement can be performed in the mass spectrometer cell in many ways, such as by Time Of Flight (TOF) or quadrupoles, for example. In an MS1 experiment, various methods like High-Energy Collision Dissociation (HCD), Collision-Induced Dissociation (CID) or Electron-Transfer Dissociation (ETD) may be used. In an MS2 experiment, depending on the selection criterion (Selected Reaction Monitoring (SRM), Data-Dependent Acquisition (DDA) or Data-Independent Acquisition (DIA)), the second round of fragmentation is usually performed by collision with neutral molecules, such as argon, helium or nitrogen. Those various methods are also often parameterised, for example HCD can be performed at various energy levels, resulting in various spectrum qualities.

Overall, the mix of technologies and their parameterisations can have a wide impact on the type and quality of spectrum generated. For example, MALDI-based ionisation mostly generates fragments that have a charge of 1+, whereas ESI-based ionisation generally generates fragments in the 2+ and 3+ charge range. The ProteomeTools synthetic peptide dataset paper (Zolg et al., 2017) reports that a normalized collision energy (NCE) of around 25 achieves better quality spectra than other settings, and this same observation is repeated in other work (Diedrich et al., 2013). The interpretation of the intensity value of peaks depends

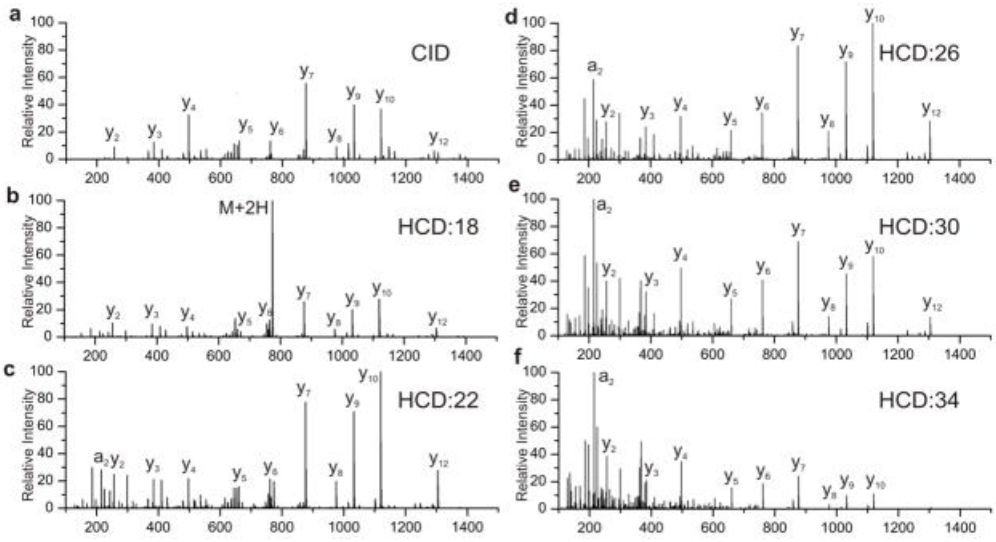


Figure 4 – Different fragmentation technologies and parameters result in differing spectra distributions for the peptide ELGQAGVDTYLQTK. **a**: CID fragmentation. **b-f** HCD fragmentation at different NCE. Image reproduced from Diedrich et al. (2013).

on various experimental parameters (Fazal, 2013), and different fragmentation methods will fragment peptides along different ion series (Zhang et al., 2013), and so forth.

1.2. Peptide Fragmentation

In light of the above, while a survey of fragmentation methods, their parameterization, and their resulting patterns is outside the scope of this work, Collision-Induced Dissociation (CID) and HCD are briefly compared here to illustrate how the choice of method once again impacts the generated data.

In Figure 4, the fragmentation patterns of CID and HCD at various energy levels are demonstrated for one example peptides. As can be observed in the Figure and as previously reported in Diedrich et al. (2013), some ion types such as internal ions, i.e. fragments that do not contain either terminus, are specific to HCD fragmentation. Moreover, there is energy-specificity in the distribution of ions: internal and immunonium ions (low-mass amino acid markers useful to identify the presence of specific amino acids and help resolve mass conflicts) begin to appear at higher energy levels. In addition, the intensity of high mass peaks tends to go down as collisional energy increases, while the intensity of low mass peaks increases in the same direction. Between CID and HCD, HCD results in higher-resolution spectra at the cost of acquisition speed (Jedrychowski et al., 2011).

Figure 3B illustrates an MS2 mass spectrometry run assuming HPLC fractionation, HCD fragmentation, inert gas collision MS2 and quadrupole collision cell and mass filters and so forth (i.e. a QExactive mass spectrometer), because this matches the instruments used in

Name	Type	Mass Range	Resolution	Accuracy
Waters LCT Premier	TOF	18 000	10 000	2-5
Agilent LC/MSD TOF	TOF	7 000	10 000	2-5
Bruker MicroTOF	TOF	3 000	10 000	2-5
Waters QTOF Ultima	QTOF	17 500	32 000	2-5
Waters QTOF Micro	QTOF	30 000	5 000	2-5
MDS Sciex Qstar XL	QTOF	20 000/40 000	10 000	2-5
Bruker BioTOF	QTOF	10 000	20 000	2-5
Bruker Apex IV	FT-MS	66 000	100 000	<1
Ion Spec	FT-MS	18 000	1 300 000	1
Thermo LTQ FT	FT-MS	2 000	500 000	2
Thermo Q Exactive Plus MS	Orbitrap	6 000	140 000	<1

Table 3 – List of common mass spectrometry instruments with their resolution (FWHM), accuracy (ppm) and mass range (m/z). FWHM: Full width at half maximum, i.e. the m/z of a peak divided by the width of the peak at half the maximum intensity for that peak.

the generation of the datasets used in this work (Zolg et al., 2017; Hebert et al., 2014; Wang et al., 2018).

Beside the far-ranging impact of the slew of potential technical choices that may be made in the data generation process, other properties of the instruments used to perform the analysis, such as accuracy across mass ranges and resolution, also have important repercussions for peptide identification software: methods that work in low-resolution routinely work poorly for high-resolution data, and vice versa. Resolution is defined as the ability to separate ions at a given m/z, i.e. how closely two peaks can be and still be differentiated; while accuracy relates to the ability to measure said m/z, i.e. how close the reported and true m/z for a peak are. Higher-accuracy instruments help resolve nearly-isobaric masses to the right amino acid, such as distinguishing between asparagine (with a mass of 114.04293) and aspartic acid (with a mass of 115.02694). An instrument with higher resolution allows for distinguishing those same two amino acids instead of having to combine their signal.

In Table 3, the accuracy and resolution of common instruments is indicated.

As shown in the table, the choice of instrument will vary the accuracy and resolution of the captured data. This has downstream implications on the choice and output of identification software. In Kilpatrick and Kilpatrick (2017), the authors outline how the mass calculation for amino acid sequences can veer off of accurate values as the accumulation of small mass errors at the amino acid level, derived from calculations based on lower-accuracy masses, can shift longer sequence theoretical masses by values of the order of 5 Da. Additionally, using software that expects high resolution data to analyze low resolution data will result in poor quality identifications due to a lack of well-defined high-resolution mass patterns for precise match scoring. On the flipside, using software designed for low-accuracy data with a high accuracy mass spectrum will also result in poor performance as the additional high

PTM Class	Mass Difference	Amino Acid
Methylation	14.0157	R, K, E, M, H
Phosphorylation	79.9663	R, K, S, T, Y, H
Oxidation	15.9949	M, M, C, K, E, Y, W, H
Acetylation	42.0106	C, K, S, W, Y
Glycation	Variable	Variable
Lipidation	Variable	C

Table 4 – List of PTM classes, their respective amino acids, and their mass difference respective of this amino acid. Only single-modifications are shown for simplicity, however modifications can apply multiple times to the same amino acid. For example, lysine can be double- or triple-methylated. We refer the interested reader to Kitamura and Galligan (2023) for a more detailed review of common (human) PTMs

resolution peaks can't be taken into account and will instead act as noise against the scoring algorithm and its reference database.

1.3. Post-Translational Modifications

In addition to protocol variations, peptides may also undergo post-translational modifications (PTMs), which are chemical modifications to the peptide's amino acids. Common PTM classes are shown in Table 4.

Modifications can be applied as part of the experimental protocol. For example, carbamidomethyl cysteine is often used to prevent cysteine residues from forming disulfide bonds between or within proteins. When such modifications are applied, they are referred to as "static", since they are universally present, as opposed to "dynamic" modifications, which may or may not be present on an amino acid.

Since modifications shift the mass of amino acids, they can cause further mass conflicts. In Kim et al. (2016), the authors found that lower accuracy instruments like ion-trap mass spectrometers would fail to distinguish about half the modifications in the UniMod database. On the other hand, a high accuracy instrument with a 1 ppm mass accuracy would be able to distinguish almost all non-isobaric modifications. More than 10% of modifications are isobaric.

1.4. Identification Software

Peptide identification software paradigms largely fall into one of two categories: de novo sequencing software, and database-driven sequencing software. Examples of de novo sequencing software include NovoHMM (Fischer et al., 2005), PepNovo (Frank and Pevzner, 2005), PNovo (Chi et al., 2010), DeepNovo (Tran et al., 2017) GraphNovo (Lu and Chen, 2003), and QuasiNovo (Cleveland, 2013), among many others. Database software include PeaksDB (Zhang et al., 2011), Andromeda (Cox et al., 2011), Comet (Eng et al., 2012) (an

open version of the earlier SEQUEST (Eng et al., 1994)), X!Tandem (Craig and Beavis, 2004), Mascot (Perkins et al., 1999), and plenty more. Generally, de novo methods have not been particularly successful in the unconstrained case. In constrained scenarios, e.g. when the exact sequence is not known but its structure and length are constrained, some methods have shown some success (Bhatia et al., 2012). As such, database-driven methods are a lot more popular for most experimental settings. Despite their success, identification rates in real datasets using database-driven methods are far from perfect. For example, Michalski et al. (2011) report that a significant amount of peptide species are not detectable with common mass spectrometry analysis software. Generally, in a non-synthetic sample, it is common to detect around 50% of peptides only (Hebert et al., 2014).

The following text describes the main concepts in database-driven peptide identification pipelines. In Chapter 2, we present a new database-driven search engine and explain in more details the operations in such software.

1.4.1. Spectrum Databse. Spectrum databases are essential for database-driven peptide identification: they are the database that are searched against the experimental spectra provided as input queries in an identification pipeline. Two major approaches are currently in use to generate this database: experimental spectra from peptides that have already been generated or identified, and in-silico database generation.

Spectral Libraries. In general, searches against spectral libraries result in higher sensitivity Deutsch et al. (2018). However, spectral libraries may be incomplete Deutsch et al. (2018) and thus less desirable than in-silico digested searches against generated spectra in some appllcations. Moreover, as previously discussed, the various choice dimensions in experimental designs result in different fragment distributions in spectra, thus not all libraries may be suitable for all experiments.

Sequence Libraries. The primary alternative approach is to generate spectra from in-silico digested peptide sequences. This increases the complexity of the identification pipeline greatly at several levels: the simulated digestion may not match the stochastic distribution of real-life digestion products even with well-known and well-behaved enzymes like trypsin, classical spectrum generation methods do not generate overly realistic spectra (although novel methods address this issue, if but in a subset of settings) Gessulat et al. (2019); Tiwary et al. (2019); Liu et al. (2020), and more variables are introduced in the software pipeline requiring care with statistical analysis, and tuning for optimal peptide retrieval performance.

Implications for Software. Spectrum library and sequence library software have typically operated by similar mechanisms with slight, but notable, differences. Likely the most popular approach for spcetrum library search is normalized dot-product-based scoring methods Griss (2016). At their most simple, these methods compute a dot product or analog

between a spectrum to search for and a reference library of spectra of known peptides. However, care must be taken to account for bias due to potentially overwhelming contribution of a few high-intensity peaks to the score, or the statistical distribution of dot products vs the score obtained for a spectrum pair, as done in SpectraST Lam et al. (2007, 2008); Shao et al. (2013). These methods also do not readily allow for open PTM searches, although extensions to these algorithms endow them with the ability to perform such searches Ma and Lam (2014). Other posstible extensions include spectrum clustering to allow identifications from unlabeled mass spectra Önder et al. (2014), and projected dot product for chimeric spectrum resolution as in M-Split (Wang et al., 2010).

For spectrum generation, correlation-based approaches such as Comet Eng et al. (2012) or X!Tandem Craig and Beavis (2004) are usual. Some approaches attempt to provide probability-like scores (like X!Tandem, which corresponds to an inverse likelihood, or Mascot Perkins et al. (1999), the first engine to offer a probabilistically interpretable scoring function). Match quality evaluation metrics like p-value or q-value Käll et al. (2008) can be more readily applied to these scoring algorithms to obtain relevance estimate for predicted match scores.

Unlike for spectral libraries, sequence libraries also require care around source proteome selection, digestion rule, and spectrum generation: algorithms must be properly parameterized and implemented so as to allow for the modeling of events like missed cleavages, partial digestion, probabilistic amino acid-based digestion affinity, and more or less complete sets of digestion rules. Yet, it is common for digestion algorithms to only implement in trypsin, for example, digestion before K or R but not when preceded by P. Some software like ExPASy's PeptideCutter Gasteiger (2003) support more complete rules, but such software often does not support the full scope of potential digestion modeling options simultaneously. Those parameters are also often tuned differently per experiment, including such parameters as variable and static modification generation, with some rare tools like IdentiPy Levitsky et al. (2018) offering automatic tuning of a subset of these parameters. Once peptide sequences are generated, the spectra can be generated from these sequences. Modern, deep-learning driven approaches like DeepMass Tiwary et al. (2019), Prosit Gessulat et al. (2019) or Predfull Liu et al. (2020) can achieve very high correlations between predicted mass spectra and experimental spectra for the same peptide sequence, modifications and charge in a subset of cases. More classical approaches may generate major ion series such as the complete y-, and b-series, taking into account the target experimental parameters and their characteristics. Intensity prediction in these methods is typically not available Gessulat et al. (2019); Tiwary et al. (2019). Feature support varies widely, with some tools only generating the main series, while others are able to generate immonium ions, internal fragmentation, neutral losses, etc., although adding more predited peaks can increase the amount of spurious matches against

the database and reduce overall recall rate, again often requiring manual parameter tuning for each experiment.

A related problem is the generation of decoy peptides, which is equally critical for the target-decoy approach to false discovery (TDA-FDR)-based evaluation of search results. Databases generated by in-silico digestion, and hybrid or deep learning-driven approaches, can naturally generate decoys using the same methodology as they generate target peptides, and the quality of the decoy database thus depend almost entirely on the quality of decoy peptide sequence selection (so as to satisfy the prerequisite conditions for valid TDA-FDR evaluation (Elias and Gygi, 2007)). On the other hand, decoy generation for spectral libraries generated from inventoried experimental spectra is more complicated, and work on improving the situation is relatively recent (Lam et al., 2009). Alternative approaches like probabilistic approaches implemented in SpectraST Shao et al. (2013) offer a decoy-free evaluation method that can potentially outperform decoy-based approaches in some cases.

In the TDA-FDR error correction method, peptides that should not exist in the sample being analyzed (decoy peptides) is submitted to the same spectrum generation process as used for those peptides possibly in the sample (target peptides). Under several regularity assumptions which are reviewed more thoroughly in Chapter 4, including the assumption that the likelihood of an identification on the wrong peptide for a given spectrum is equal to the likelihood of identifying a peptide in the decoy database, one can pose $\texttt{FDR} = \frac{N_{\text{incorrect}}}{N_{\text{correct}}} = \frac{N_{\text{decoy}}}{N_{\text{target}}} = \frac{N_{\text{decoy}}}{N_{\text{incorrect+correct}}}$ Elias and Gygi (2007). Many alternative formulations and attempted corrections to this estimation method exist. Some common variations add 1 to the numerator leveraging different sets of assumptions, while others suggest using $\frac{2N_{\text{decoy}}}{N_{\text{decoy}}+N_{\text{target}}}$, but there is little evidence in the wild for a universally superior method and the choice of formula remains debated Jeong et al. (2012).

For decoy database generation, common methods can generally be categorized as protein-level, peptide-level, and distributional, that is: methods that generate decoys based on transformation of whole source protein sequences, of peptide sequences, or from modeling amino acid distributions. Examples in the first category include reversing protein sequences and then performing in-silico digestion, or randomly shuffling amino acids along a protein sequence before performing digestion. Similar methods can be used at the peptide level, i.e. after in-silico digestion. Finally, distributional methods such as deep learning, hidden markov model, or de bruijn methods try to model amino acid distributions and generate new sequences without other basis from the initial sequences Lee et al. (2022). Variants of these methods, such as performing protein reverse decoy generation but keeping K and R amino acids in place (because trypsin digests before these amino acids), e.g.. "pseudo-reverse" methods, are sometimes used as well. In general, there is evidence that the protein-reverse method is the least biased Jeong et al. (2012), although choice of method remains varried in the literature.

1.4.2. Query Processing. Input MS/MS spectra may be very noisy for various reasons, including the presence of contaminants, calibration issues, experimental noise, and more. To achieve confident peptide identification, it is important to properly filter the spectrum to improve its signal to noise ratio. To do so, many strategies exist, the most common of which are probably intensity normalization as performed in common tools like Comet and X!Tandem (Eng et al., 2012; Craig and Beavis, 2004), peak count-based spectrum removal (Levitsky et al., 2018) and de-isotoping (Levitsky et al., 2018; Teo et al., 2020) as in IdentiPy and others, and wavelet-based noise peak removal such as with MSConvert (French et al., 2014; Kessner et al., 2008). The choice, parameterization, and effect of these transformations differ based on search engine and preprocessing choices and is usually left up to the user. To the best of our knowledge, there has not been any comprehensive evaluation of the effect of these filtering methods outside tool-specific evaluation relying on final identification rates, or comparable metrics. In particular, deep learning-driven methods are well known, in general, to deal well with large, noisy datasets. With the advent of deep learning-based algorithms to improve various parts of the software proteomics workflow, the suitability of these transforms (i.e. both the question of whether those filtering methods should be applied at all, and if not, which transforms are most appropriate, as well as which parameters thereof are suitable in this new paradigms) may not match their suitability in the more historically standard pipeline.

1.4.3. Scoring. Scoring is arguably the most important part in a peptide search engine. It relates a candidate peptide (via its spectrum as per the database used in the software) to an experimental spectrum. Many scoring methods have been proposed, like those of OMSSA, Comet, X!Tandem, and X!Tandem's K-Score (a precursor to comet's) (Geer et al., 2004; Eng et al., 2012; Craig and Beavis, 2004; MacLean et al., 2006), and many are proprietary and only partially publicly explained, like PeaksDB's or Andromeda's (Ma et al., 2003; Cox et al., 2011). Pioneered by Mascot (Perkins et al., 1999), an important development is that of probabilistic scores, which allows the easy computation of quality metrics such as p-values and E-values (Eng et al., 2012) to better guide hit retention. Scoring functions typically combine both intensity and difference in m/z between expected and experimental peaks, as in Comet and X!Tandem (Havilio et al., 2003; Eng et al., 2012; Craig and Beavis, 2004). Deep learning methods like PredFull (Liu et al., 2020) and Prosit (Gessulat et al., 2019) have been proposed for intensity prediction (or full-spectrum prediction in the case of PredFull), and while they achieve accurate predictions for those quantities, no algorithm is currently able to leverage them at the scoring level. Indeed, while Prosit was introduced into Andromeda recently (Wilhelm et al., 2021), the authors could only generate an additional match quality metric after the completion of the normal Andromeda search process, and

provide this metric to Percolator for rescoring [1]. Scoring is complicated by many factors, including degraded performance based on spectrum library size in light of evaluation metrics like TDA-FDR-based evaluation (for example, due to the count of one-hit wonders, i.e. rare incorrect identifications that score extremely well, increasing as library size increases) (Yen et al., 2011). For illustration purposes, the popular Comet algorithm (which is also used in Sequest and in X!Tandem with the k-score plugin) is presented below:

1.4.4. Rescoring. Rescoring is a very popular strategy that increases peptide identification at a given FDR threshold, however the tendency to treat it as a blackbox is a mistake (Gupta et al., 2011a). Methods like Percolator (Käll et al., 2007; Spivak et al., 2009; Granholm et al., 2012) and others like PeptideProphet (Ma et al., 2012) are very popular, especially the former. Percolator trains a support vector machine (SVM, described in the machine learning section below in more details), which is a machine learning model that finds optimal weighing factors for each input feature according the the maximum separating hyperplane for the samples. However, the feature given as input to percolator can be completely arbitrary and are typically (but not always, which is even more dangerous) generated by the developer of a search engine that wants to leverage percolator for rescoring. Without careful evaluation of the effect of the choice of features, it could leak information to percolator, allowing it to enrich for false hits (i.e. targets that are not the true peptide represented in the spectrum), exactly as described in Gupta et al. (2011a). We discuss this in more details in Chapter 4.

1.4.5. Protein Inference. In a typical shotgun mass spectrometry experiment where the goal is to identify proteins in a sample of interest, once peptides have been found, further processing is required to finish the identification task: to resolve the collection of peptides found previously into a set of proteins. While the prevailing paradigm is to use either a probabilistic model relating the count of distinct peptides relating to the same protein to the likelihood this protein was in the sample (Eng et al., 2012; Ma et al., 2003), or simply to only select proteins with 2 or more peptide hits, the latter approach has been argued to be brittle and likely inappropriate (Gupta and Pevzner, 2009), while the former may be a major reason why protein-level FDR control tends to fail harder than at the peptide level (Jeong et al., 2012).

Protein inference is a very complex topic that is outside the scope of this work. We focus instead on only the problem of peptide identification for various reasons, including the lack of similarly high-quality data for whole-protein identities, complications related to error control and common heuristics Gupta and Pevzner (2009); Wu et al. (2018); Bogdanow et al. (2016), and the exponential complications involved in dealing first with novel peptide identification methods and second the choice of protein-level algorithms to identify proteins from peptides.

1. According to personal communications, the authors tried to use Prosit as part of the scoring process proper but ultimately failed

Another reason we focus on peptide identifications is because improvements in peptide identifications typically correlate strongly with identification performance for proteins Zhang et al. (2015), as we confirm in our limited measurements.

Common strategies for protein inference include the two peptide rule, a popular but contested method Gupta and Pevzner (2009); The et al. (2016), Fisher's method of combining p-values, the best-peptide approach, and the posterior error probability multiplication method. The interested reader is directed to The et al. (2016) for a more complete overview of protein inference considerations with percolator.

The two peptide rule states that a protein is identified when two peptides pass false discovery selection and match the same protein. This approach aims to solve the one-hit wonder problem, that is the identification of proteins by incorrect single peptide matches, by requiring a more stingent match criterion to call a protein. However, previous work has cast doubt on the validity of the technique Gupta and Pevzner (2009); The et al. (2016), and evidence that one-hit wonders truly can be observed suggest this rule may serve to remove correct hits more often than thought (Gupta et al., 2007).

In the Fisher method, the Fisher Combined Probability Test computes the statistic $X^2 = -2\sum_{i=1}^{k} \log p_i$ where p_i is the p-value of the i^{th} identification, which is chi-square distributed under some regularity assumptions. This aims to reduce one-hit wonders by penalizing identifications leading to many spurious matches and shares the same basic idea as the multiplication of peptide posterior error probability method, which simply takes the product of all peptide posterior error probabilities for a given protein (in this case, spurious hits will have a posterior error probability near 1, thus controlling for one-hit wonders). Both methods also share the same objection: the requisite independence assumption between the peptide identifications is dubious The et al. (2016).

Finally, the best peptide approach takes only the best single peptide identification for a protein when considering protein identifications. In Savitski et al. (2015), the authors found that selecting the best-scoring PSM as the representative for the protein (thus using the peptide's identity as decoy or target for protein identification and error control) performs best in a small dataset. This result was reproduced in The et al. (2016) and is the method implemented in Percolator.

1.4.6. Evaluation. Since in typical experiments, there is no robust method to know what a sample contains, peptide identification quality is assessed using ad-hoc approaches, such as thresholds on the score provided by identification software (Zolg et al., 2017; Cox et al., 2011). Currently, the most popular quality control approach is the target decoy approach to false discovery rate estimation (TDA-FDR) (Elias and Gygi, 2007), however there is an important body of literature demonstrating that this method is not appropriate: on one hand, an obsession for maximizing identification rates at a fixed false discovery rate threshold has

caused software developers and scientists to bypass the basic assumptions underlying FDR estimation (covered later), thus rendering TDA-FDR estimation ineffective (and typically greatly underestimated) (Gupta et al., 2011a; Wang et al., 2015; Hubler et al., 2019), on the other, there are systemic weaknesses in the use of TDA-FDR for quality assessment, including the ad-hoc decision to choose a threshold (compare the choices made in papers such as Hebert et al. (2014); Gupta et al. (2011a); Laumont et al. (2016); Zolg et al. (2017)), the different mathematical models implying different FDR values at the same decoy/target ratios (Balgley et al., 2007a; Käll et al., 2008; Elias and Gygi, 2007), the limitations this approach implies through the assumptions underlying the mathematical model, and so forth (Jagannadham et al., 2021; Couté et al., 2020; Martiánez-Bartolomé et al., 2008; Wang et al., 2015; Hubler et al., 2019). This makes comparison of various search engines impossible in practice, since the identification count at a so-called fixed FDR threshold actually represents a completely different threshold for each engine due to chronic misestimation (Yuan et al., 2014; Gupta et al., 2011a; Kandasamy et al., 2009; Välikangas et al., 2017; Jeong et al., 2012).

Additionally, these methods can be biased based on how the decoy database is generated and how the search is performed (e.g. separate search vs combined, reversed peptide sequence decoys vs random protein sequence decoys, and many other factors) (Gupta et al., 2011a; Jeong et al., 2012).

While the TDA-FDR quality control strategy is often used both at the peptide and protein level, beyond the pitfalls described above, it has also been suggested that TDA-FDR control at the peptide level is more reliable than at the protein level (Jeong et al., 2012; Elias and Gygi, 2009).

We believe evaluation is a major weakness in the field and a significant hurdle to further software development for peptide identification. Evaluation and its issues, especially in light of the common modern paradigm of applying rescoring methods on top of a first pass search, is the subject of Chapter 4.

2. Machine Learning & Deep Learning

Deep learning algorithms are a subset of the machine learning class of algorithms differentiated by the depth of the architecture involved (LeCun et al., 2015; Schmidhuber, 2015). Therefore, it is sensible to cover more general, basic concepts in machine learning before tackling deep learning-specific topics. In the following, we cover the notions of "learning", "inference" (which, in this context, will be used interchangeably with other terms, such as prediction, discrimination, generation or classification depending on context) and the evaluation of models in the context of machine learning, briefly mention a select few relevant

machine learning algorithms, and finally take a more focused, deeper dive into essential deep learning building blocks.

2.1. Machine Learning

Machine learning describes a class of algorithms which differ from others in that instead of describing program behavior, they instead describe how to adapt values in the program that parameterize program behavior. While various experts have different ideas of what the first example of machine learning is (LeCun et al., 2015; Schmidhuber, 2015) (in part because there is no consensus about where statistical methods end and machine learning begins (Bzdok et al., 2018; LeCun et al., 2015; Schmidhuber, 2015)). The three important steps in machine learning development are (1) learning; (2) inference; and (3) evaluation.

"Learning" is the process of finding values that parameterize the model so as to achieve some objective, such as maximizing classification performance, or minimize regression error. Various algorithms are used for different models, as is discussed with some examples in the below section. In order to ensure the model does not overfit the data, it is important to have multiple subsets of the data for training and for the evaluation of the model's ability to generalize. Many strategies exist, such as cross-validation, bootstrapping, and various set splitting strategies (Xu and Goodacre, 2018). In deep learning, it is almost universal practice to split the dataset in use in three parts: a training set, which is used to train the model, a validation set, which is used to ensure the model is not overfitting the subset of the data it is using for training, and is often the basis for hyperparameter optimization, and a test set, which is used to verify the actual performance of the model once the optimization of parameters and hyperparameters is complete (James et al., 2013). This common practice stems from the large amounts of data required to train deep learning models: since the dataset is so large, the model typically does not have a harder time converging to a solution from the split procedure, as opposed to the regime where little data is available. In this latter scenario, the cross-validation approach is favored.

Cross-validation involves splitting the dataset the same way as described in the three-way split for deep learning, however only two splits are performed: one for the development, or training, set, and one for the validation set. Then, the split is repeated [2], the model is retrained on the new train set, and reevaluated on the new validation set. The measurements from the repeated training-evaluation workflow can then be used to assess how well the model generalizes, and how sensitive it is to the exact training examples in the dataset. The literature suggests that the fewer examples are retained in the validation set, the better (i.e. N-split cross-validation for a dataset of N examples is optimal) in terms of controlling for overfitting, at least in the few samples case (Larsen and Goutte, 1999). It is often

2. Depending on strategy, this can be with (less common) or without (more usual) replacements, for example

reported that 5-10 splits work well in practice (Arlot and Celisse, 2010), while using too few splits, such as in 3-way cross-validation, can cause significant performance misestimation (and therefore the misinterpretation of a poorly generalizing model). 10-way cross-validation is a very common choice among practitioners, so as to retain good quality generalization while minimizing compute requirements. Overall, the rule of thumb seems to be: 5-way cross-validation at minimum, more is better, with 10-way cross-validation a good default choice.

We use the term "inference" loosely to refer to "obtaining an output from the model". For example, "inference" for a classification model consists of using the model to predict the class of some input, while "inference" for a generative model consists of generating some data using the model.

There are several evaluation strategies, such as simply evaluating on a held-out test set, or reserving a "gold set", referring to a well-curated dataset meant to be representative of the task of interest. Correct evaluation can be complicated, as it is not always easy to completely separate training and evaluation data: if the data has strong dependency between input features, then even though two input vectors (one in the training data and one in the evaluation data) are identical for a few features but different for the rest, the trained model may still achieve high performance simply from having overfit on those few common features, but this begs the question of how much overlap is too much. More concretely, consider the case of peptide detection in a mass spectrum. Mass spectra among replicates, even for synthetic peptide samples, can vary widely (Liu et al., 2020; Gessulat et al., 2019), due to various factors such as slight variations in environmental conditions, instrument precision, contamination, and more. In this case, it is likely wiser to completely separate the sets based on the peptides, i.e. the label of interest, rather than the input, i.e. the feature vectors. However, even then, it is not necessarily clear if that is sufficient separation, since there may be strong correlation between the spectra of the peptides PEPTID and EEPTIDE.

2.1.1. Common Algorithms. Three popular machine learning algorithms (excluding deep learning) include random forests, support vector machines (SVM) (Boser et al., 1992), and hidden markov models (HMM).

SVMs attempt to find the "best" hyperplane that optimally separates two sets of points, where the "best" line is the one that maximizes the distance between the closest exemplars of each set, i.e. it finds the optimal margin classifier. The classical algorithm consists of solving a set of primal/dual optimization problems via quadratic programming, although the more commonly used algorithm as of the time of writing is the Sequential Minimal Optimization algorithm first presented in Platt (1998). SVMs only possess a solution in the case of a linearly separable dataset, in which case they provide a decision boundary suitable for classification between two classes. Extensions exist to adapt SVMs in multi-class contexts (Hsu and Lin, 2002), and applying a kernel to the data allows the use of SVMs for non-linear

classification (in this setting, the resulting algorithm is often called "kernel SVM", even though this setting was already discussed in the original svm paper by Boser, et al. (Boser et al., 1992)). SVM-based machine learning algorithms have been used in proteomics, such as to perform rescoring in Käll et al. (2007), which is the most popular rescoring algorithm in current use.

Decision trees are not so commonly used on their own in modern machine learning workflows, but they are a necessary element of the popular random forests, described below. Decision trees follow a tree-like model to identify interactions between input variates. At a high level, decision trees are inferred from splitting the hypothesis space recursively according to some criterion, such as parsimony or entropy. Common algorithms for decision tree inference include ID3, CD4.5 and CART. Inference then proceeds by selecting appropriate tree branches at each decision points based on the new input datum to be classified or regressed Rokach and Maimon (2005).

Random forests are an ensemble method that pools the predictions from many individual decision trees (Breiman, 2001). There exists various ways to organize the overall learning procedure. One option is to subset the input data and to assign each subset to one tree to learn from. Another is to randomize the original data's output before presentation to each tree (Breiman, 2001). Typically, result aggregation (i.e. from the prediction of individual trees into a single prediction for the entire forest) is performed by majority vote (Breiman, 2001). The subletting and potentially stochastic nature of the algorithm greatly helps in reducing overhitting risk inherent to tree-based algorithms (Breiman, 2001). Along with SVMs, random forests are popular, general-purpose classification algorithms. Random forests have been used in such applications as MHC-I affinity prediction (Boehm et al., 2019), and Pepid (presented in Chapter 2) features an optional random forest-based rescoring algorithm.

Hidden markov models (Baum and Petrie, 1966) are machine learning algorithms that model sequences that are assumed to be markov, i.e. such that the value at a given timestep can be generated from only the single previous value, without dependencies on the rest of the value history. HMMs assume an inference chain through time of latent variables, that is, hidden states (states which are not known from the data), and mutually independent observed variables which are generated from the corresponding hidden state at the given timestep. HMMs owe their names from the process being modeled assuming a markov transition process over hidden states (which is inferred from observed states). A hidden markov model are ultimately defined by the distribution functions $g(h_{t+1}|h_t)$, called transition probabilities, which is the distribution function from which transitions are generated from the current to the next hidden state, and $f(x_t|h_t)$, the emission probability distribution from which outputs are generated given the current hidden state. The viterbi algorithm (Viterbi, 1967) is commonly used to infer those distributions, which allows a trained HMM to be

quite efficient. Algorithms such as NovoHMM (Fischer et al., 2005) use HMMs for de novo sequencing of mass spectra.

2.2. Deep Learning

As a subclass of machine learning, deep learning algorithms are also defined by the concept of "learning", with the main axis of differentiation with broader machine learning approaches being the depth of the models involved. While many types of deep learning models have been proposed (from deep belief networks (Hinton et al., 2006) to deep boltzmann machines (Salakhutdinov and Hinton, 2009) or even earlier work such as Ivakhnenko (1971)), modern approaches have converged on a few common points:

(1) Complete models are composed from basic building blocks;

(2) Training is driven by gradient descent;

(3) Models are organized in neural networks using hierarchical "layers" of independent "neurons";

(4) "neurons" in a layer implement the simple affine transform followed by a non-linearity (proving the full layer activation function form $\sigma(W\mathbf{x} + \mathbf{b})$);

Due to their pervasive importance as building blocks of virtually all modern deep learning models, convolutional neural networks (CNNs), fully connected neural networks (FNNs) and recurrent neural networks (RNNs) are covered in their respective subsections below. While various algorithms have been described for training neural networks, from hebbian learning (Journé et al., 2023; Hopfield, 1982) to contrastive divergence (Carreira-Perpiñán and Hinton, 2005), or various derivative-free approaches (Rios and Sahinidis, 2012), gradient-based learning remains a mainstay due to the efficiency of this learning technique in practice. A brief discussion of the main advantages and disadvantages of gradient-based learning is provided below.

The hierarchical, layer-wise, independent-neurons organization of modern deep learning neural networks and the simple function of each neuron likely derive in large parts from computational simplicity, especially due to matrix multiplications and additions on stream processors such as GPUs being so efficient (able to achieve speedups over CPU implementations on the order of 50x (Schmidhuber, 2015), but relying on the problem being highly regular for representation as a set of matrix operations.

Although deep learning is far less sensitive to the quality of the data preprocessing pipeline, a good choice of data representation can still have significant impact on a model's learning speed and capacity requirements. Figure 5 shows two common representations for a mass spectrum (peak list and vector of m/z bins) followed by intensity value range normalisation, which are common choices for machine- and deep-learning based pipelines taking mass spectra as input (Gessulat et al., 2019; Tiwary et al., 2019; Tran et al., 2017).

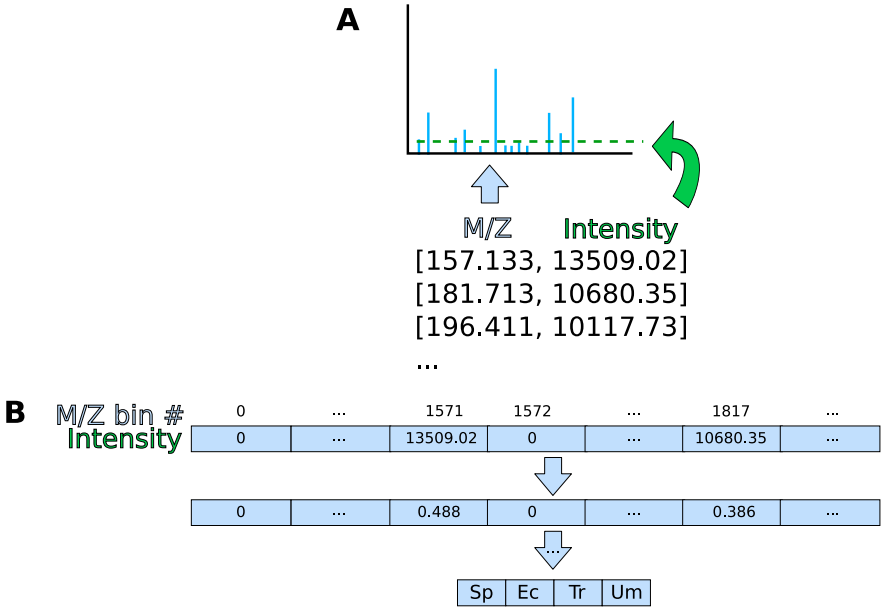


Figure 5 – Data, such as a mass spectrum (displayed in **A**), can be represented in multiple ways (such as a list of peaks and intensities, or as a vector of intensities where the index in the vector corresponds to a m/z bin, here presented in 0.1 Da increments). Proper representation (**A** to **B** top) and preprocessing (**B** top to bottom, with the abstract final SpEcTrUm representation) can vastly increase a model's ability to learn.

2.2.1. Gradient-based optimization. Deep learning models are almost exclusively trained (equivalently, "learn", or "are optimized") using gradient-based algorithms. The most basic such algorithm is stochastic gradient descent (Kiefer and Wolfowitz, 1952), where instead of taking the gradient with regard to the entire dataset (corresponding to an "epoch"), a subset (i.e. a batch) of examples is used at a time (although some authors would argue that it is more accurately called batch gradient descent to differentiate between gradients estimated from single examples or minibatches of examples (Khirirat et al., 2017)). This formulation is useful due to the potentially very large amount of data that can be used to train those models vs the reality of computing resource limitations, as well as due to the stochastic effect of batch selection on the gradient descent path helping the optimization process "escape" local minima (Kleinberg et al., 2018). In practice, the ADAM algorithm (Kingma and Ba, 2014) is likely the most common algorithm choice due to its good performance and its relative insensitivity to its hyperparameters (Kingma and Ba, 2014). Overall, weight updates take the form $w := w - \eta \frac{\partial L}{\partial w}$ where η is a learning rate hyperparameter, L is the loss function, or objective that the model is minimizing, and w is a weight. This can be efficiently computed due to the overall model architecture using the gradient chain rule.

2.2.2. Building blocks. For all intents and purposes, all modern deep learning-based neural networks are defined in terms of just three basic building blocks: the convolutional neural network (CNN), the fully-connected neural network (FNN) and the recurrent neural network

(RNN). Those building blocks are presented in Figure 6. While modern paradigms such as associative memory (Gulcehre et al., 2017), attention (Gregor et al., 2015; Vaswani et al., 2017) and skip connections (Srivastava et al., 2015) are commonly used in most large architectures, those patterns are themselves simply implemented using the same basic building blocks described here.

FNNs are simple stacked layers of neurons where each neuron in a given layer is connected to all neurons in the layer immediately above. They are also called feedforward neural networks. When they possess at least one hidden layer, they can also be referred to as multi-layer perceptrons (MLPs) (Haykin, 1994). They are most suitable for data on which no further structural prior (e.g. spatial codependence) is expected. They can learn input vector position-specific patterns (such as "if the first unit is above the value 10, then output a positive value"), but aren't inherently capable of position-equivariance, which can make them suboptimal for the analysis of data with positional patterns where the pattern may equivalently appear anywhere in the input (e.g. images or sound data). For efficiency reasons, FNNs are expressed as matrix operations (i.e. where each column represents the weights corresponding to a connected neuron, and each row consists of a feature vector as input, along with the addition of a bias vector), which easily dispatch to stream processors for maximum throughput.

RNNs (Rumelhart et al., 1986), of which the most popular renditions are the Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and the Gated Recurrent Units (GRU) (Cho et al., 2014), are specifically suited for sequence processing. Conceptually, they are FNNs that input and output two feature sets: a "state" that is updated as the input sequence is processed, and the proper input, are passed to the initial layers, while the output is composed of a local, or temporary output, along with an updated value for the state. This conceptual FNN is then repeatedly applied through the elements of the sequence. RNNs can be seen as a extension, or generalization, of HMMs. Due to the temporal dependencies during training and inference, RNNs tend to be slower than both CNNs (see below) and FNNs, but they are capable of learning relations that are theoretically unlimited in time (Hochreiter and Schmidhuber, 1997) (although that does not necessarily hold in practice (Li et al., 2023)), as opposed to CNNs (limited by kernel size) and can be shown to be turing complete (Siegelmann and Sontag, 1992). A common variation is to operate two independent RNNs, one across the input sequence and one in reverse. This is often referred to as a BiRNN (for bidirectional RNN), especially BiLSTM or BiGRU when LSTMs or GRUs are used as the RNN.

CNNs (LeCun et al., 1989, 1998) are similar to FNNs, except they replace the matrix multiplication with a convolution (which is often implemented as a matrix multiplication in fourier space (Frigo and Johnson, 2002)). They are position-equivariant by design, but require suitable data and capacity to be able to learn sufficiently complete features to achieve

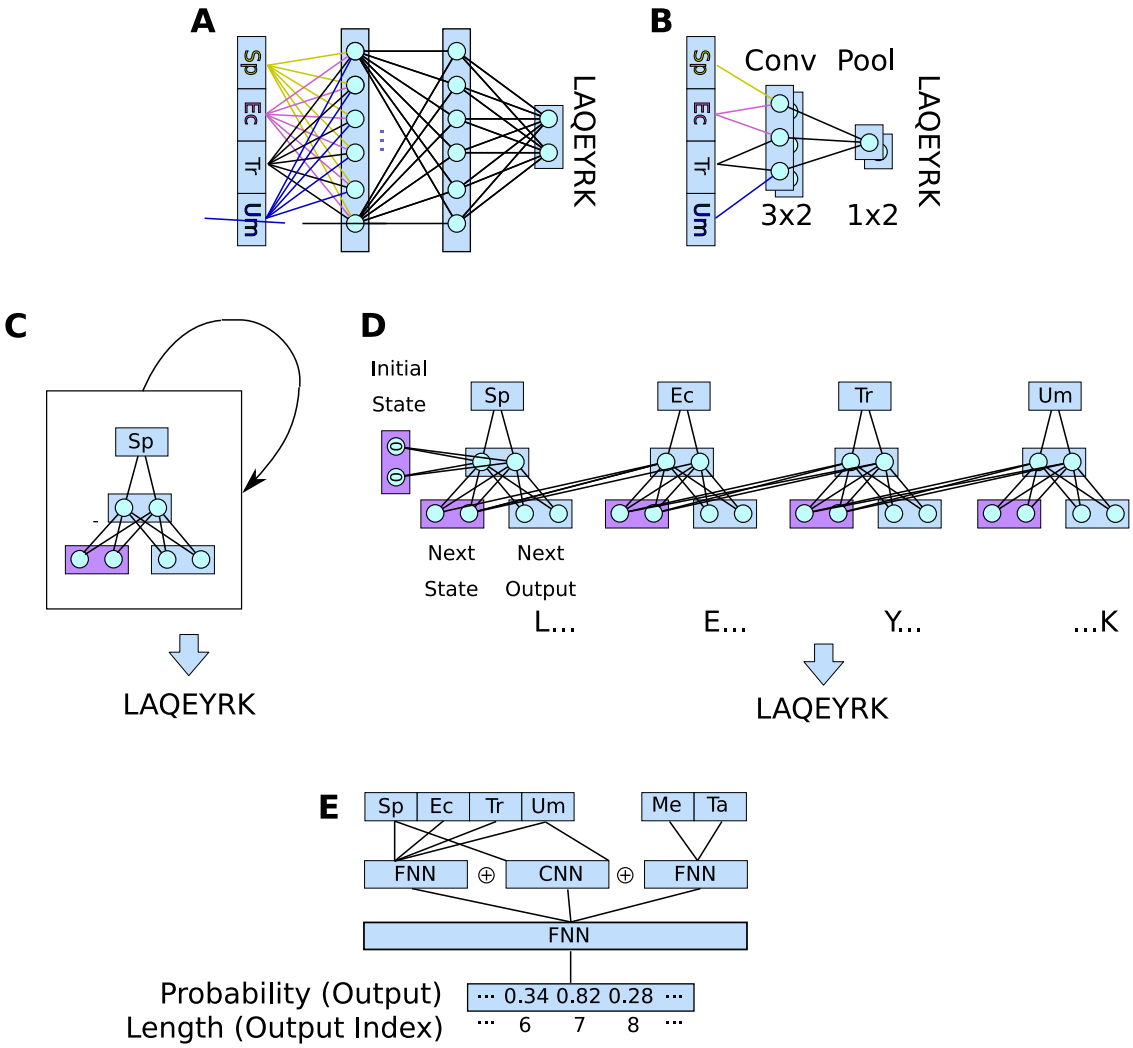


Figure 6 – Illustration of basic neural network building blocks that form the foundation of virtually all modern deep learning neural networks, and of a combined architecture. In these examples, a mass spectrum (SpEcTrUm), as processed in Figure 5, is the input of each neural network. **A**: FNN. Each connector represents a link between an input (left) and output (right) following the basic equation $o = \sigma(wi + b)$ for some activation σ, weight w and bias b. **B**: CNN. While various designs are possible, iterating convolutional (Conv) and pooling (Pool) layers is a common design. Convolutional layers' kernels can be stacked to alter output dimensions (from 1 to 2 in the example). Pooling layers apply a function such as the maximum over a window in the input (the entire input in this example). **C** A single-step diagram of an RNN, the state (pink) from each timestep is carried to the next and the output is accumulated to obtain the final prediction. **D** The same RNN unrolled through time, i.e. across the spectrum. **E**: A high-level diagram of the length prediction model described in this work. A FNN processing metadata (MeTa) as well as another FNN processing the spectrum and a CNN processing the same spectrum are combined by concatenation of their output, which is then processed through additional FNNs to output length probabilities, where the probability that the spectrum represents a peptide of length L is output at index L in the vector.

scale- or rotation-equivariance (commonly achieved by performing data augmentation on the input data (Shorten and Khoshgoftaar, 2019)).

Complete neural networks are usually composed of one or multiple of those building blocks, as illustrated in Figure 6D, which presents the architecture of the peptide length prediction model which is described in Chapter 3.

2.3. Activation Functions

While the choice of activation function typically only results in minute differences in training convergence efficiency and inference-time performance when the model is properly designed taking input data into account, some activation functions have seen more general success than others. An interesting case is that of the logistic sigmoid, once the most popular general-purpose activation function, now relegated for use only in output layers. Cited reasons include issues with saturation: the output of a logistic, defined as $\frac{1}{1+e^{-x}}$, at an input of 0 is of 0.5, the logistic of 0.5 is 0.622, and so forth, thus in "deep" models, 0 values can saturate at higher layers, preventing good learning. Its immediate popular successor has been the hyperbolic tangeant sigmoid, defined as $\frac{e^x-e^{-x}}{e^x+e^{-x}}$, which does not saturate about 0 and ranges between -1 and 1. More recently, the rectified linear activation (ReLU), defined as $\texttt{max}(x, t)$ for a threshold t (typically held at 0), has become the de facto standard activation, owing to its efficient hardware implementation and good practical performance during optimization, despite the nonexistence of gradients at t (although this does not generally affect gradient-based learning, which can simply use an arbitrary subgradient instead). Many variants of ReLU have been proposed, but their use over plain ReLU remain debated.

Beside general-purpose activation functions and the logistic sigmoid used for probability-like outputs, another common activation function is the softmax, defined as $\frac{e^{x_i}}{\sum_j e^{x_j}}$ for an output vector x. The softmax transforms output values into a probability-like vector and is the standard activation function used for categorical outputs, where output elements can then readily be interpreted as the probability of the input being in each class as represented by the output vector.

3. Proteomics Meets Machine Learning

Over the years, many machine learning algorithms have been proposed to solve various aspects of identification proteomics, from HMM-based de novo sequencing (Fischer et al., 2005) to SVM-based rescoring (Käll et al., 2007) or intensity prediction using random forests (Degroeve and Martens, 2013). More recently, many deep learning models have shown success in various parts of the proteomics workflow (Guo et al., 2016; Bengio et al., 2013), efficiently predicting such varied properties as retention time (Giese et al., 2021), to mhc compatibility (Andreatta and Nielsen, 2015), or intensity prediction (Gessulat et al., 2019; Tiwary et al.,

2019) to full spectrum prediction (Liu et al., 2020), among many others (Zeng et al., 2022; Tran et al., 2017; Wen et al., 2020; Meyer, 2021; Schoenholz et al., 2018b). Unfortunately, nearly none have ever been applied to full, standard mass spectrometry workflows, let alone in a reusable way. Exceptions include Prosit, which has been integrated into Andromeda (Wilhelm et al., 2021), and the de novo models that are part of PeaksDB searches (Ma et al., 2003; Tran et al., 2017). Meanwhile, rescoring, almost always by Percolator (although other tools like PeptideProphet Ma et al. (2012), the linear discriminant analysis method in Du et al. (2008) or the non-parametric method of Zhang et al. (2008), have also been proposed and used in some cases), has become the de facto standard in proteomics identification pipelines, despite the rather large body of work, as described in Section 1.4.6, showing concerning systemic issues with this paradigm. We believe the main reason why so few of these apparently groundbreaking machine learning and deep learning tools have been integrated into commonly used pipelines is due to the relative complexity of performing this integration, on both a technical (existing identification tools are designed as if they are meant to operate as a blackbox method on top of wetlab experiments, which affects how difficult modifying and extending them can be, as opposed to if they were designed as tools (or set of tools) that should be adapted for the current experiments; this seems to lead to the development of new tools for each experiment class instead of extensions to existing methods) and social (licenses and source availability, for example) levels. We address this by introducing a drylab research-oriented peptide search engine in Chapter 2. We also propose a novel view on deep learning tool development for proteomics and combine a new deep learning model with our proposed search engine in Chapter 3, showing the practicality of the proposed search engine's architecture and the power of deep learning on realistic identification workloads.

To better showcase how machine learning is used in practice to improve proteomics, we go over Prosit (Gessulat et al., 2019) and Percolator (Käll et al., 2007) in detail below. While both methods are flawed, they nevertheless exemplify full processing pipelines for different parts of the mass spectrometry-based identification workflow. Prosit was successfully integrated in a real identification software solution by using percolator, and rescoring by percolator is nowadays a standard step of most identification pipelines, and those methods bear relevant to the rest of this work.

3.1. Percolator

Percolator went through various changes since its original version (Käll et al., 2007; Spivak et al., 2009; Granholm et al., 2012). In this section, we describe the latest iteration as of this writing (namely the version as of Granholm et al. (2012), although we deliberately avoid discussion of new development in the codebase that have not been analyzed for validity

yet). Percolator's central component is the SVM machine learning algorithm: provided a set of PSMs as input into its custom Percolator INput (PIN) format, Percolator attempts to find optimal weights for each of the search engine- or user-computed features from which the input file was generated, so as to best discriminate between decoys and targets (illustrated in Figure 7. The SVM ultimately provides a distance between the separating hyperplane and the point of interest, which is an increasingly positive value for inputs that the model believes more strongly to be targets, or an increasingly negative value for inputs that the model believes to be decoys.

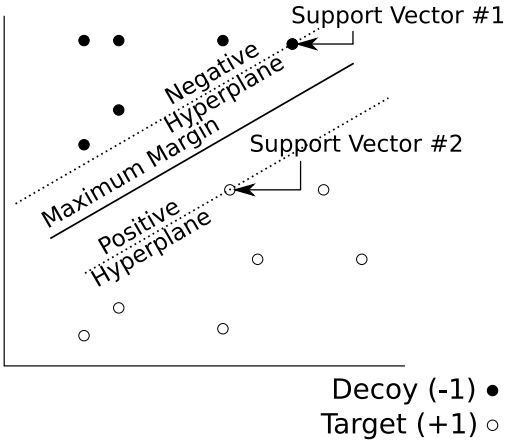


Figure 7 – The central piece of the Percolator rescoring system, an SVM that separates decoys from targets. The SVM identifies the maximum margin hyperplane based on the positive and negative hyperplaned defined by the support vectors (the feature vectors that are closest to the separating plane). The distance to the maximum margin hyperplane indicates the separation "confidence", as elements closest to the hyperplane are harder to separate than those further away. For Percolator, decoys are labeled as -1 (hence, any negative distance to the hyperplane marks a predicted decoy)

In an attempt to avoid overfitting, Percolator operates with a 3-fold cross-validation approach, proceeding as follows:

(1) The dataset is split in 3 parts;

(2) Percolator sequentially selects one of the 3 parts to serve as a validation set, and combines the remaining 2 parts to serve as a training set;

(3) The SVM is trained on the training set, and the optimal parameters are saved.

Since the optimum in each subset may differ, which would cause the predicted "decoy-ness" or "targetness" from the model to differ for the same input depending on the specific SVM used, Percolator performs a "normalization" adjustment to the output of each SVM to try to make the outputs comparable across the models. Finally, Percolator uses the SVM corresponding to the held-out set each point corresponds to in order to rescore it as per the scoring-normalization pipeline. The adjustment method consists of setting the score

threshold for 1% q-value (an alternative measure of FDR with similar values and properties (Käll et al., 2008)) as the target 0 value on a line, with the median decoy PSM score as -1 for the same line, then projecting the score values unto the line within each fold in the cross-validation. The overall process is illustrated in Figure 8

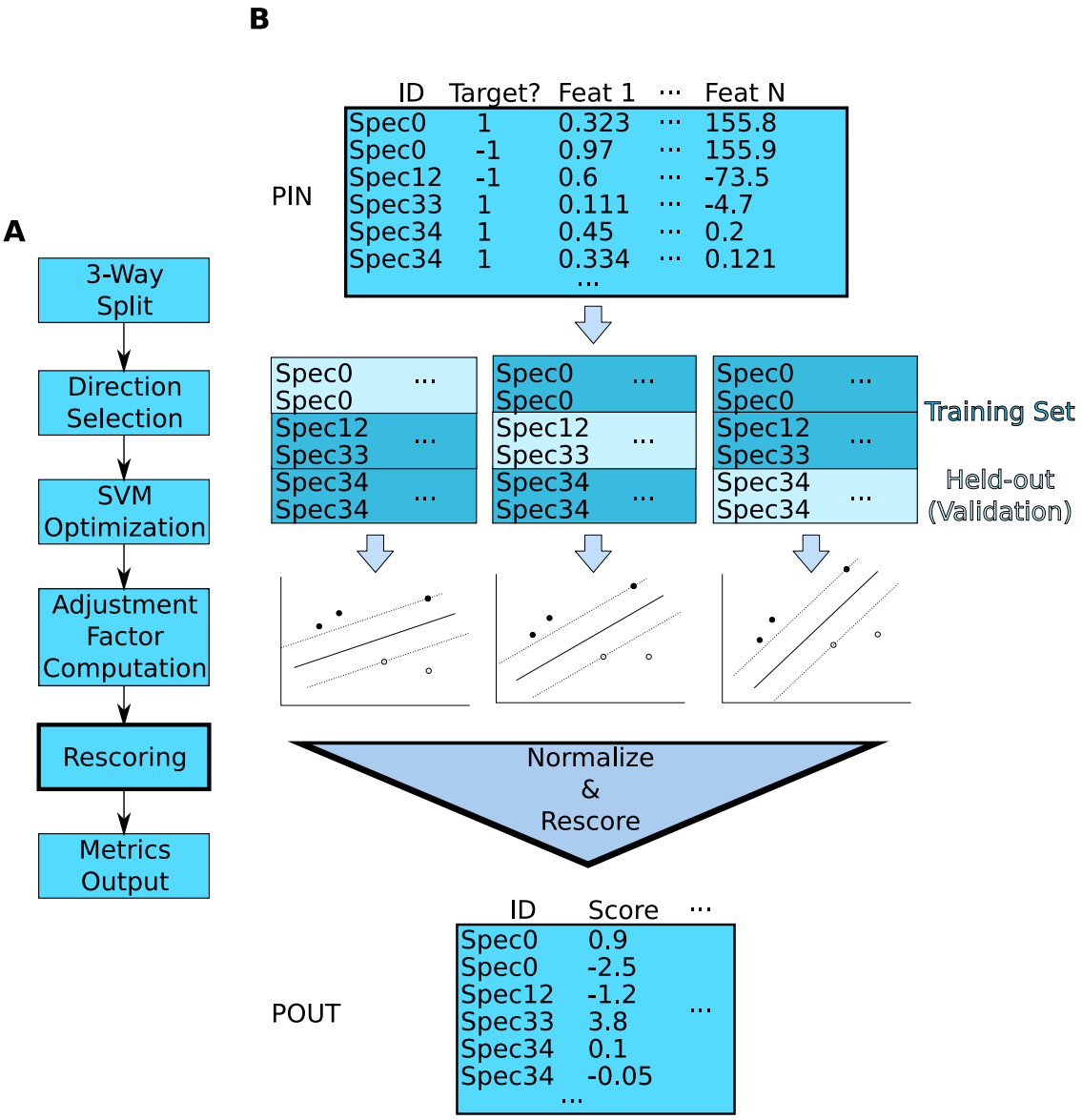


Figure 8 – Overall process for Percolator rescoring. **A**: The high-level steps. **B**: Illustration of the most important steps in the pipeline.

The initial direction selection consists of finding the best input feature as computed by the amount of targets identified at 1% q-value (or whatever target is set in the settings by the user). Percolator performs this step to determine which feature to use as the "score" feature (instead of being provided the engine score feature name explicitly). This is used as a sanity check to verify that the SVM training process does not fail beyond a certain level as well

as to determine the labels for the training process (i.e. PSMs passing the q-value threshold on the basis of this "score" feature form the "positive" examples, while all decoys – not the PSMs failing below the threshold – are "negative" examples). After the first iteration, Percolator uses the accumulated normalized SVM weights so far to compute new scores for the data, using those new scores to identify the next positive and negative sets to use.

3.2. Prosit

Prosit uses a deep learning model to predict both indexed retention time (iRT) and the intensity of b- and y-series ions given an input sequence. The iRT prediction depends only on the sequence, but for intensity prediction, other features, such as the normalized collision energy (NCE) and the precursor charge, are also provided to the model. The architecture is a pretty straightforward and classical arrangement of deep learning building blocks as presented previously, and is illustrated in Figure 9. Briefly, the sequence part is processed by a neural network composed of an embedding layer (equivalent to a specially structured fully-connected layer), two BiGRU layers, and an attention layer (i.e. an arrangement of fully-connected layers). The metadata processing part takes the concatenated metadata (namely the NCE and charge) and processes it through a fully-connected network. The outputs from these two parts are elementwise-multiplied together, after which another BiGRU layer takes the resulting features as input. Finally, a fully connected network is applied to each timestep in the output sequence of the last BiGRU to produce the final fragment intensity value corresponding to each fragmentation point in the input sequence.

As discussed previously, processing the input data correctly is important for efficient learning. To this end, Prosit uses fairly standard representations for its inputs: the sequence is represented as an integer index sequence, where each integer is the index in an array of amino acids. The sequence is padded with zeros after the sequence is encoded, to ensure all sequences in a minibatch are of the same (maximum) length. For the metadata, the NCE is presented as a normalized value between 0 and 1, while the charge is a one-hot encoding, that is a vector of values containing a 1 at the index representing the charge for this instance, and zeros everywhere else.

The model outputs intensity estimates only for y- and b-series ions, and for all charge levels up to the precursor charge level, inclusive.

To integrate the Prosit intensity predictions as part of a full database-driven search workflow, the authors first perform a standard database search with the usual theoretical spectrum generation facilities, then compute the prosit predictions for the retained PSMs only and calculate the quality of the match between the prosit-generated spectrum and the experimental spectrum in the PSM. The match quality value is provided to Percolator as a

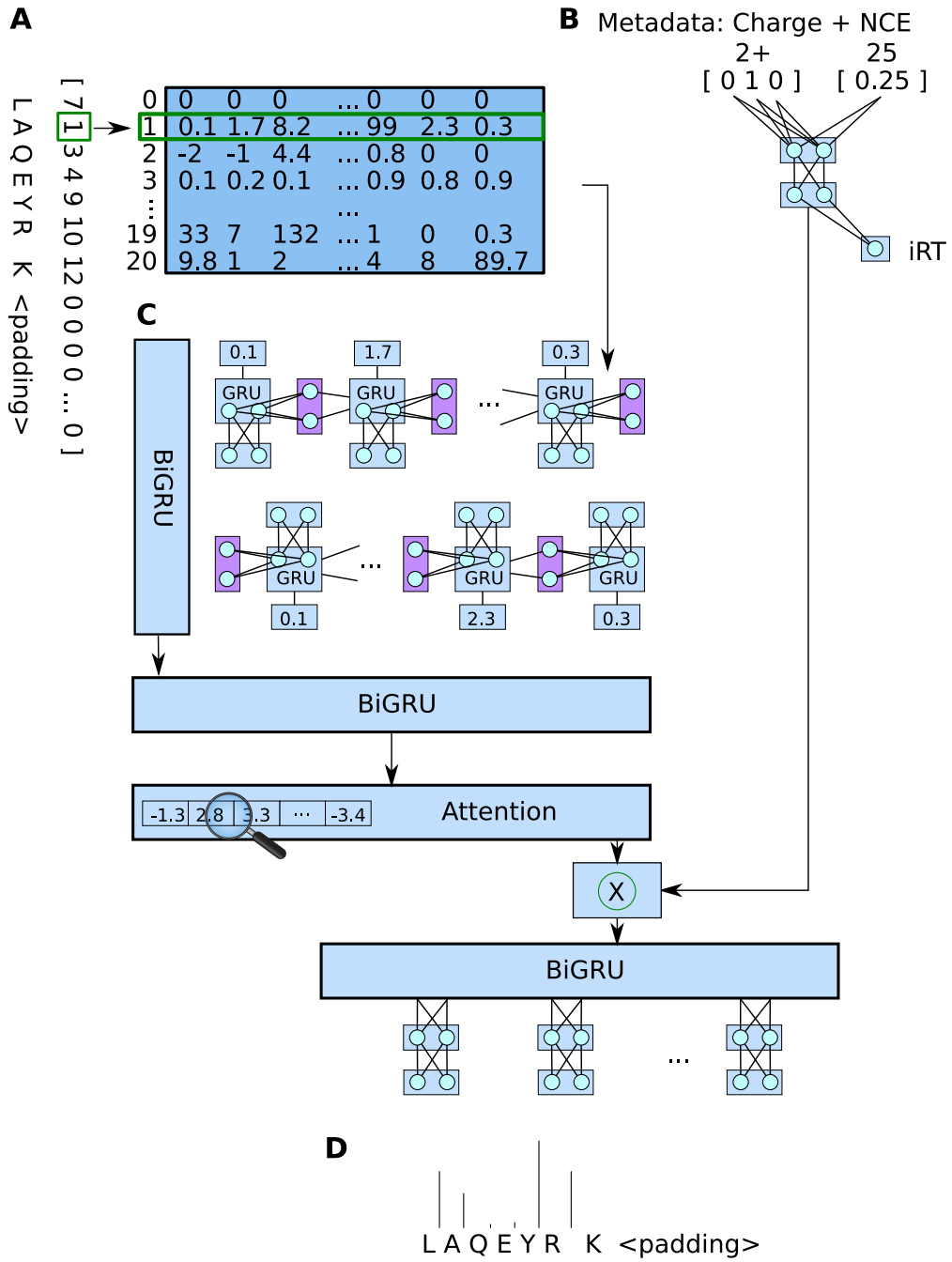


Figure 9 – Illustrated Prosit model. **A**: The sequence, encoded as a vector of amino acid indices, is converted via an embedding layer that converts each entry into an optimized vector. **B**: The metadata is processed by a simple FNN to output iRT. The output of the model (before the iRT output layer) is also used as part of the intensity prediction model segment. **C**: The embedded sequence is processed by layers of BiGRUs, illustrated once here. An attention layer attends to the output sequence from the BiGRU, after which the result is elementwise-multiplied with the output of the iRT prediction model's hidden layer. Finally, the multiplied vectors are processed by another BiGRU, followed by a FNN applied individually to each output. **D**: The final output: b- and y-ion intensities for each valid charge level, corresponding to the fragmentation between each amino acid pair.

feature, and percolator performs rescoring by relying on this metric to better separate decoys from targets.

Prosit is a good illustration of the power of deep learning: the model does not use any particular tricks or tuning specific to the data (only to the data type, e.g. by encoding charge as one-hot and so forth), and especially not to the experiment, i.e. the same architecture can readily be reused on other datasets, for example for different fragmentation type, charge range, length range, etc. Despite this flexibility, the model achieved state-of-the-art results at the time. Similarly, our Pepid search engine (in Chapter 2) integrates a full-spectrum prediction model (that is, it predicts both the m/z and intensity of peaks instead of being limited to known series like Prosit) as an optional and experimental module based on a standard CNN-based architecture.

The work presented in this thesis started with a series of deep learning-centric data exploration experiments. These experiments eventually revolved around concepts in de novo sequencing, due to the inherent lack of required scaffolding (such as peptide databases and non-data-driven algorithms for their processing), the existence of previous work in end-to-end machine learning and deep learning approaches for de novo sequencing, and the presence of good ground truth data in the forms of datasets like ProteomeTools and Massive-KB, discussed later in the text. These experiments culminated in the discovery that deep learning can predict the length of a peptide present in a spectrum, and that a model trained this way generalizes across datasets and even species. Even more impressively, these predictions can readily be used to improve database-driven peptide identification. This is the topic of Chapter 3. The original manuscript was based on combining those predictions with the IdentiPy search engine in an attempt to minimize the risk of software error, and to speed up the research process (peptide search engines can be fairly complicated software, as they need to balance execution throughput and identification performance, and may have to process very large datasets that do not fit in memory, among other concerns. They are also sensitive to slight programming errors. Indeed, we discovered a critical flaw in the IdentiPy scoring function that nevertheless only had minor identification implications simply because the peptide mass generation function in IdentiPy was coincidentally such that the bug triggered mostly in less important regions of the spectra. This bug was later corrected when the IdentiPy authors introduced a "fast" version of their scoring function, but it is not clear if they noticed the bug in the original version or if the bug was fixed by coincidence as well).

IdentiPy was chosen for four main reasons: it promised higher identification performance than other engines, it was open source and therefore easily modifiable, it was specifically designed with some customization features like pluggable user functions, and it was written in python, which allowed ready integration of our deep learning models. Unfortunately, we found, during the process of modifying IdentiPy for combination with our length predictor, that the inherent design of this tool (mostly based on one-to-one processing of query-candidate pairs maximizing in-memory processing) was not particularly amenable to the efficient execution of deep learning primitives. In addition, the modification process was fairly complicated beyond having to change the processing paradigm from "one-to-one" to batch-oriented because many of the components' processing functions are designed with the other steps in mind. Surveying other open source engines, such as x!tandem, comet, morpheus and ms-gf+, we found that none featured a mix of suitable customizability for faster research iteration (as the pluggable functions in IdentiPy putatively allow, or to a much lesser extent the pluggable score in x!tandem), batch-oriented design (as is available in comet and x!tandem), and controls on the output of PSMs (to be able to analyze results for debugging and function development). In particular, it appears that current peptide search engines are concerned only with outputting identifications, as opposed to facilitating

research in bioinformatics methods for the identification of peptides. Therefore, Chapter 2 is concerned with the development of a more suitable search engine called Pepid, which is concerned primarily with the development of drylab methods for peptide identifications. Pepid features competitive identification rates while being faster than common search engines in its basic configuration. To showcase its ease of use for research, we develop a new machine learning-based rescoring algorithm as well as a deep learning-based full spectrum prediction algorithm, which are also discussed in that chapter. We show that combining these tools, as well as the length prediction algorithm from Chapter 3, allows significantly more peptide identifications than the baseline configuration and other search engines. These results from combining deep learning with a classical database search approach were made possible in large parts (in fact, almost exclusively given the relative difficulty of doing the same with other engines in our experience) due to the batch-oriented, phase-by-phase design of our engine, and the easily pluggable functions across its pipeline.

A common pain point encountered in the work discussed in the previous chapters is evaluation. We found that the typical target-decoy based false discovery rate-controlled identification count metric used in the literature can be inconsistent with the true identification count as measurable when ground truth peptide identities are available (such as in the ProteomeTools and Massive-KB datasets), which we already show without discussion in Chapter 2. Indeed, there is a notable body of work in the literature evaluating and discussing this discrepancy. In Chapter 4, we explore this issue once more, especially in light of the current paradigm of the use of rescoring (in particular the Percolator algorithm) as a post-processing step to any and all peptide identification pipelines. We point out that the identification-based metric invalidates cross-search-engine comparisons, that rescoring algorithms inherently bypass the required assumptions for FDR calculation, and demonstrate experimentally that this effect is notable in practice. We conclude that any comparison between search quality should, at present, use a dataset containing ground truth peptides at least for "result calibration" purposes as we are aware of no existing method that properly controls for incorrect evaluation of FDR, or may control false discovery rate, with or without the use of decoys.

Chapter 2

Pepid: a Highly Modifiable, Bioinformatics-Oriented Peptide Search Engine

Jeremie Zumer, *Institute for Research in Immunology and Cancer*
Sebastien Lemieux, *Institute for Research in Immunology and Cancer, Department of Biochemistry, Université de Montréal, Montréal, Québec H3C 3J7, Canada*

Publication

This article in preparation will be submitted to a journal to be determined at a later date.

Contributions

Author contributions to this article are as follow:
— Jeremie Zumer performed all other work.
— Sebastien Lemieux advised and corrected the authoring of this paper.

Abstract

Current peptide search engines are optimized for wet-lab workflows, i.e. they operate in an "end-to-end" manner to achieve good identification results, not to be modified or provide algorithmic insight. This makes developing new software methods to solve problems in peptide identification methods difficult, often requiring a full engine rewrite. Recently, many deep learning methods were proposed as solutions to various parts of the peptide identification task, but virtually none of those methods have been implemented in any actual peptide search process. We believe that the lack of a reliable bioinformatics research platform

for peptide identification that enables such integrations is slowing down proteomics research as a whole.

Indeed, peptide identification from the software point of view is a series of complicated steps, and decisions made at one step may have critical influence in the next step. For example, the choice of candidate generation method for use in the TDA-FDR evaluation methodology will impact the relative count of decoys vs. targets in the search database, which may bias the accuracy of the FDR evaluation, for which methods have been proposed to address this issue both at the candidate generation step or at the FDR calculation step (Elias and Gygi, 2009; Wang et al., 2015; Hubler et al., 2019; Jeong et al., 2012).

Meanwhile, the lack of properly engineered engine to make software modifications painless has prevented the ready integration of modern, deep learning-driven tools into search pipelines, making evaluation of those methods on practical workflows nigh impossible and forcing developers and researchers interested in using these methods to implement a new search engine altogether, or to leverage extensive software development skills to restructure an existing engine for which both source code and license are amenable to modifications and, when desired, publication of results.

We present pepid, a bioinformatics research-oriented peptide search engine. Unlike other search engines, pepid is specifically designed with ease of computational research in mind. Our design is highly flexible and allows easy modifications with little required software development expertise, allowing researchers to focus on analysing and improving peptide identification methods. It also takes recent computational trends into account, such as the recent slew of deep learning publications in proteomics, and features a multi-phased batched operations design that is more appropriate than the spectrum batch "end-to-end" designs of existing search engines for those approaches. We show that pepid is competitive with common engines in terms of both identification rates and runtime, forming a minimum required baseline to enable further identification research.

The design of Pepid, which emphasizes modularity, operates in a computationally efficient way on the batch level,and retains intermediary information for evaluation and reuse, have been instrumental for the development of the follow-up chapter on length predictions, whose improvements in peptide identifications have also been evaluated in the results presented in this paper.

Pepid is available as open source software under the MIT license at `https://github.com/lemieux-lab/pepid`. **Keywords** Peptide Identification, Search Engine, Proteomics

2.1. Introduction

Peptide identifcation is a technically and scientifically difficult problem both on the wet- and dry-lab side. Current peptide search engines are designed for "end-to-end" operations, i.e. to provide a streamlined process for proteomics experiments considering wet-lab users, with non-separable operations from spectrum and protein processing all the way to scoring and output. This has the unintended consequence of making computational research in peptide identification harder, because deviating from the established paradigms effectively requires rewriting the entire engine. Recently, many deep-learning algorithms have been proposed to solve various problems relevant to peptide identification, such as theoretical spectrum generation (Gessulat et al., 2019; Liu et al., 2020), retention time prediction (Zeng et al., 2022; Giese et al., 2021), peptide-spectrum scoring, and so on (we direct interested readers to more comprehensive reviews of deep learning methods in proteomics, such as Wen et al. (2020) and Meyer (2021)). Unfortunately, these methods cannot be readily tested on top of most labs' software workflow because making the requisite changes requires good software development expertise, as well as time and financial budget that may not always seem like a good use of resources to most labs. As a result, it is not clear which of these methods, if any, is actually useful for database-driven peptide searches, and even in the rare cases where those methods are implemented with an existing search engine (Wilhelm et al., 2021), it is not readily possible to port those changes to another engine to benefit from a proven method because two different engines may operate in totally different ways.

State-of-the-art peptide-centric search engines for peptide identifications can be classified in many different ways. One classification axis relevant to the difficulties in modifying engines for actual use with novel techniques is their license status (i.e. proprietary or not), and source status (i.e. closed vs. open, published vs. unpublished). Breakthrough deep learning algorithms have been proposed, but they cannot be added to proprietary search engines and released legally (except by such search engines' authors themselves), slowing down research and making convincing experiments to demonstrated added value on top of some commonly used engines difficult. In addition, current open-source search engines do not have deep learning methods in mind and are designed in such a way that integrating machine learning models to their pipeline is difficult: for example, deep learning algorithms are designed for efficient batch processing, potentially at one specific stage of the process, whereas current engines are desgined to perform all processing steps across groups of inputs.

In this paper, we propose Pepid: a new, open-source search engine that is designed from the ground up with dry-lab research and deep learning methods in mind and, in general, to further research and development of tools and algorithms for peptide identification. Our search engine features a multi-phase, batched pipeline design. Each step in the pipeline is fully optional, uses configurable user function attachments, and supports a variety of user

settings to customize computational schedules. We believe this is a more appropriate design for applications such as deep learning than the usual "end-to-end" approach of other options, at the cost of incurring some overhead that we show to be negligible overall.

We implement two common scoring algorithms and their extensions as well as a combination score function, and we show that Pepid is competitive in both runtime and identification rates. We believe that deep learning is already instrumental to achieving best-in-class performance in peptide identification, and that further research at the intersection of deep learning and proteomics is critical to next-generation peptide identification research. We provide experimental deep leanring methods with our source release to demonstrate how one might go about integrating such algorithms, although discussion of those details falls outside the scope of this paper.

2.2. System and methods

Existing search engines present various tradeoffs based on their implementation and design. We discuss a select few common examples and describe how Pepid addresses flaws in current choices. The selected engines showcase some orthogonal design characteristics that make them suboptimal for Pepid's usecase (research into computational methods for peptide identification, as opposed to maximum identification rates or counts, and incidentally, throughput).

2.2.1. IdentiPy

IdentiPy (Levitsky et al., 2018) is an open-source search engine written in python 2.7, using cython in parts to improve performance. Its main feature is its high configurability, providing users with custom extension points allowing staff with limited technical skills to provide and apply arbitrary python functions at limited parts of the pipeline. This paradigm also means the intermediary computation steps are exposed to the user, allowing more flexibility in how the pipeline operates and runs. It makes use of multiple processors via multi-processing and performs task dispatch on the unit spectrum level (that is, each query spectrum is individually processed by the next available process on a first-come-first-served basis with no task batching).

However, only a small set of extension points are supported, and the single task dispatch design means that operations with short computations and long preparation times (e.g. those leveraging GPUs for general-purpose computing, as would be desired for a deep learning-driven workflow, or those that should load data from the disk in chunks) limits what kind of functions can feasibly be implemented this way due to wallclock time concerns. Moreover, it is implemented in Python 2.7, which is incompatible with Python 3.0+ and is deprecated as of January 1st, 2020; conversion is not difficult but remains a burden on potential users. It

also operates fully in-memory, which, combined with the multi-processing paradigm, makes it susceptible to the "memory explosion" problem – that is, as the task schedule is unpredictable and may use large amounts of memory, potentially exceeding available memory; this may cause the entire run to fail. It also means that the database size used in the search, as well as the query set used in any one run, are artificially restricted based on hardware availability.

2.2.2. X!Tandem

X!Tandem (Craig and Beavis, 2004) is an open-source version of the proprietary University of Washington's SEQUEST engine (Eng et al., 1994). X!Tandem is reasonably fast and efficacious, however it is designed to operate "end-to-end" for identifications, meaning there is no way to halt the process at arbitrary points. On the other hand, X!Tandem supports custom scores via a "score plugin" system. An example of a popular score plugin is the "k-score", which is the same basic scoring method that was later implemented in Comet (Tabb, 2015).

Score plugins in X!Tandem are developed in C++, which requires a certain technical sophistication from the programmer as well as providing a rather slow development cycle when working on iteratively improving the score function. By comparison, languages such as Python, Julia or Lua, which may feature slower operation at runtime, but provide better facilities for rapid application development, may be more appropriate for computational research purposes, since a better optimized version of the algorithm may replace the prototype after a novel approach is found to be advantageous.

X!Tandem also does not allow outputting more than the single best-scoring peptide match for each spectrum except when applying a filter on the score (e.g. there is no way to output an unfiltered best-10 score), which makes disambiguation with potentially close-in-quality matches impossible without further manipulation of the software, and score distribution analysis only possible with sufficient hardware to store all scores on disk, as well as temporarily in memory during X!Tandem's processing. This may also affect the performance of post-processing methods like Percolator (Käll et al., 2007) as they may exploit the greater amount of available data if provided to them.

2.2.3. Comet

Comet (Eng et al., 2012) is a reputedly fast search engine whose main scoring function is a function of the inner product between the theoretical spectrum of a candidate peptide and the empirical spectrum of the query input (where both spectra are represented as fixed-length vectors of mass to charge intervals to intensity values), normalized by the average of the same score across a window around each peak window. Comet also features a so-called E-value score, which is based on a regression over the score distribution's log-histograms (i.e.

the E-value score is an ordinary least-square approximation of the estimated log-CDF of the score distribution). Similar to X!Tandem, Comet is designed for "end-to-end" operations and does not innately support any modification to its runtime procedure. Unlike X!Tandem, it also does not support a plugin system for scores or other aspects of the computations.

Comet operates in a batch-oriented way, processing a user-specifiable amount of spectra at a time, thus bounding memory requirements, potentially at the cost of processing time. This design is more amenable to be modified to support heterogeneous compute capabilities. Indeed, Tempest (Adamo and Gerber, 2016) is such a modified version of comet and can run on CPUs or GPUs using OpenCL for faster processing.

2.2.4. Others

A plethora of other search engines exist, some proprietary like PeaksDB (Zhang et al., 2011), Andromeda (Cox et al., 2011), or the Thermo Fisher version of SEQUEST (Eng et al., 1994), and some open-source, such as Morpheus (Kim and Pevzner, 2014a) or MS-GF+ (Kim and Pevzner, 2014b), to give but a few arbitrary examples. To the best of our knowledge, beside potential additional issues related to licensing and availability of source code, all commonly used search engines share an "end-to-end" focus causing similar challenges to those described in the previous sections. Selection of the above search engines for longer exposition does not imply endorsement, and was not made on the basis of subjective of objective criteria other than the following:

— All three engines are open-source, allowing examination of internals;

— IdentiPy has inbuilt user extension points;

— Comet and X!Tandem are in common use in practice.

This provides a comparison background that enables us to estimate Pepid's viability in practice (by proxy, using X!Tandem and Comet for comparison) as well as for flexibility (especially by comparison with IdentiPy, which allows the most customization to workflow among engines we are aware of).

A qualitative summary of some search engines compared to the proposed Pepid engine is provided in Table 1.

Engine	Language	Extensible	License	Source	Phased Runs	PSM Output	Speed
X!Tandem	C++	Little	Permissive	Open	None	Filtered	Fast
Comet	C++	No	Permissive	Open	None	Top N | Filtered	Very Fast
IdentiPy	Python 2.7†	Somewhat	Permissive	Open	None	Top N	Slow
SEQUEST*	C++	No	Proprietary	Closed	None	Top N | Filtered	Very Fast
PeaksDB	Java	No	Proprietary	Closed	None	Filtered	Slow
Andromeda	C#	No	Proprietary	Published	None	Filtered	Slow
Pepid	Python 3	Fully	Permissive	Open	All	Any×	Fast

Table 1 – Qualitative summary of some search engines. Source is the status of the engine's source code. Published Source means partial or snapshot source releases are available, for example accompanying a publication. Phased Runs means an engine can perform only a subset of its operations at a time and outputs artifacts that can be used to continue a run at a later time from where the engine left off. PSM Output qualifies if an engine can output only the top N outputs, or if it can filter based on score. *: SEQUEST is commonly used to refer both to the University of Washington and the Thermo Fisher versions of the software. The listing here is for the latter. †: The IdentiPy developers are in the process of updating their engine to Python 3 at the time of writing. ×: Pepid outputs artifacts containing all scores that were above 0, and provides a condensed output with the Top N peptides for each spectrum by default. This is fully user-configurable.

2.2.5. Datasets

ProteomeTools is a dataset of synthetic human peptides. The dataset was generated using synthesized peptides based on the SwissProt database (Bateman et al., 2020).

Due to corruption in the dataset (causing some of the archive files to fail to decompress), we only use the First Pool data (which is uncorrupted). We select only the higher-energy collision dissociation (HCD) data at a normalized collision energy (NCE) of 25 as this seems to produce the best quality results out of the available settings (Zolg et al., 2017).

The One Hour Yeast Proteome is a dataset of non-synthetic yeast peptides with a baseline generated by the Thermo Fisher version of the SEQUEST search engine (Hebert et al., 2014). We use the "batched" dataset, which combines the result of all the hour-long runs.

The Massive-KB dataset contains spectra from real experiments and use a best-representative and consensus identification approach to provide a high-confidence peptide identity (Wang et al., 2018). We select the "human hcd in-vivo" subset and remove cross-digestion experiments (such as HeLa Trypsin-Lysc experiments) by excluding those entires whose provenance dataset contains keywords lysc, argc, or chymotryp. This dataset segment does not contain the ProteomeTools data which is included in the "full" Massive-KB dataset.

The ProteomeTools data provides an inordinately clean baseline using artificial peptides and pools designed to minimize mass conflicts, while the yeast dataset is a realistic real-world peptide dataset. Together, they allow us to qualitatively observe Pepid's suitability in both settings, as each may provide a different yet useful setup for research purposes.

Properties for the datasets used in this work are listed in Table 2.

Dataset	**#Specs**	**#Peps**	**#Cands**	**Frag @ NCE**
Massive-KB	1 566 292	956 580	8 410 825	HCD @ 25
ProteomeTools	1 454 139	113 583	8 410 825	HCD @ 25
Yeast	400 665	47 650	2 160 144	HCD @ 30

Table 2 – Features of the datasets used to evaluate Pepid. Peptides (#Peps) are unique pepide sequences, excluding modifications. Candidates (#Cands) are the count of Pepid-generated candidate peptides in total (i.e. including modifications, missed cleavages, and so forth) for the species database. Spectra (#Specs) are the total spectra in the query dataset.

2.2.6. Evaluation

Each search engine uses a different method for evaluation and reporting: for example, classical false discovery rate vs. q-value (Käll et al., 2008), different formulae for their computation (Elias and Gygi, 2007, 2009), etc. In order to ensure comparisons can be made, we use the same external evaluation method for each search engine rather than relying on their native tools. The code for evaluation is available in the script `gen_fdr_report.py` in the Pepid source release. We compute q-value using the FDR formula in Elias and Gygi

(2007). In addition, since FDR can be untrustworthy (Gupta et al., 2011b; He et al., 2015; Elias and Gygi, 2009; Matrix Science Ltd, 2010; Danilova et al., 2019; Jeong et al., 2012), we also report the false discovery proportion (FDP) using ProteomeTools's groundtruth labels as computed using the same algorithm as for FDR.

2.2.7. Spectrum Generation

We present a spectrum generation model designed for integration into a search pipeline. Spectrum generation quality is benchmarked against Predfull, using code and model obtained on 2023-05-16, labelled as version 2022.05.19. The pretrained model was obtained from the PredFull Google Drive indicated in the PredFull github-hosted repository (`https://github.com/lkytal/PredFull`). Since this pretrained model was trained on ProteomeTools and thus may exhibit some overfitting compared to our model. The code was obtained from the master branch of the same github repository. We choose to compare to PredFull rather than alternatives (like Prosit or DeepMass) for the following reasons: first, PredFull was previously compared to other models and found to perform better even with only the theoretical peak sequence. Second, our method also performs full spectrum prediction, making the comparison more directly appropriate. Third, PredFull is easily reproducible and is still maintained, whereas other models like Prosit have not seen an update since 2018 (as of the time of writing) and can no longer be loaded with the currently available versions of existing dependencies.

2.3. Algorithm

Pepid is designed to operate in restartable, customizable phases. Each phase can be customized both by user parameters describing behavior settings (for example, how to multiprocess the program), or by custom (i.e. fully user-written), user-specified functions (the score function, the peptide candidate generation function and the rescoring function, for example, are all user-provided, with sensible defaults implemented by Pepid). Pepid uses python's dynamic capabilities to read the name of the functions provided in a configuration file and dispatches work through these functions as appropriate during operations. In addition, users may elect to perform a subset of phases before directly examining or modifying the data, and then proceeding with remaining phases.

2.3.1. Inputs and Outputs

User configuration to the pipeline is composed of a single file: a user configuration document in an INI-like format compatible with the python `configparser` module, for which a default configuration set is provided in the `data/default.cfg` file in the release distribution. The default file features explanatory comments for each configuration option to guide

modifications. The set of query spectra as specified in the configuration file must be in the Mascot Generic Format (mgf) and the database of candidate proteins must be in the protein fasta format. Pepid's output is a database of processed queries, candidate sequences (generated in-silico from the fasta input database), and peptide-spectrum matches (PSMs) as SQLite databases. An optional pipeline step also extracts the top N (for some user-selected N) matches per query spectrum and outputs them as a tab-separated value (tsv) file. We do not use PSI formats because they are flawed in various ways both for storage, regularity, and processing in particular due to their xml nature (Deutsch, 2012; Röst et al., 2015; Shah et al., 2010; Lin et al., 2005). Instead, we choose mgf as input format since it is widely supported by existing tools and that converters exist to and from other formats to mgf (for example, Proteomatic or msconvert (Kessner et al., 2008)), and output to a format similar in spirit to the default Percolator output, leveraging familiarity with what is most likely the most popular an ubiquitous peptide identification tool in current use. Afterward, the pipeline optionally runs percolator (Käll et al., 2007) or a custom random-forest-based rescoring method on the output, and can generate a graphical report about identification performance using the target-decoy approach, false discovery rate (TDA-FDR) methodology, both after and before rescoring.

2.3.2. Pipeline

The search engine features a pipeline design composed of 6 steps: Query and Database Processing, Input Postprocessing, Search, Report, Rescoring, Final Report, which we explain in more details below. The pipeline is graphically illustrated in Figure 1A.

In the Query Processing step, the input query file is processed in a more suitable format for further operations and is stored in a SQLite database. In the Database Processing step, peptides are generated from a fasta protein database. The resulting peptides are saved in a SQLite database. User-specified functions are used to process the queries as well as to generate peptide candidates.

In the Input Postprocessing step, user-defined functions are applied to the processed queries and peptides. The user is thus able to insert arbitrary data into the database.

Besides the arbitrary post-processing function, the user-specified spectrum prediction function is applied to the database candidates. The default spectrum prediction function simply generates the theoretical b and y ion series for the candidate modified sequence for each charge level indicated by the user. The output spectrum object can be an arbitrary python object and will be inserted as a binary blob into the database. Users can change the defaults by specifying the generation of other series, such as a, c, x and z in the default implementation, or any other properties like amonia or water loss through cutomization. The default settings are chosen to achieve a good mix of identifications while avoiding the

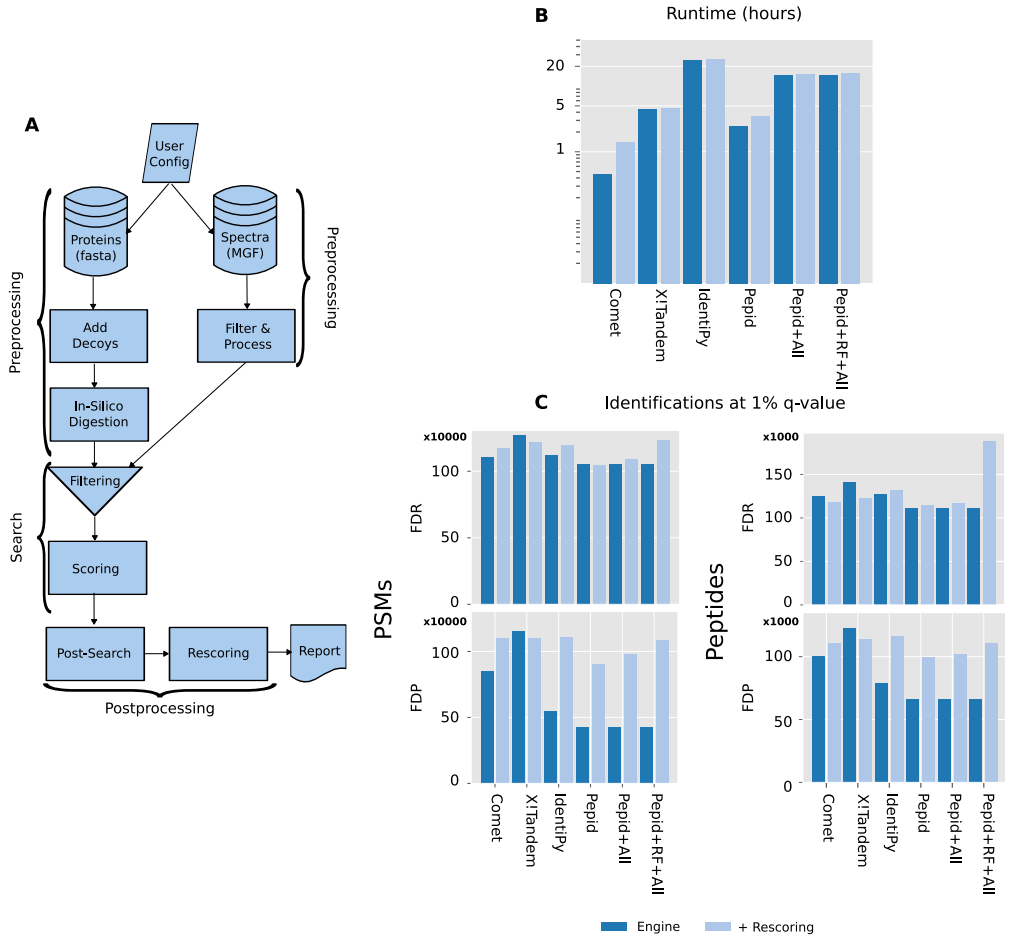


Figure 1 – Comparison between Pepid and commonly-used search engines on ProteomeTools first pool (1 458 831 spectra). RF: using the random forest-based rescorer instead of Percolator. All: using both the spectrum generator and peptide length prediction models. IdentiPy cannot process the spectra witin resource limits, so the file had to be split 10-ways, and the process run once per split. X!Tandem is only capable of outputting a single top PSM when not applying a score filter, other engines output the top 10. **A**: High-level processing pipeline steps of Pepid. **B**: Runtime performance. **C**: Count of identifications (for FDP, as per ProteomeTools baseline) at a q-value of 1% (PSMs on left, peptides on right, TDA-FDR on top, FDP on bottom).

spurious bad identifications that overloading such a generated spectrum with peak types can result in. Since the cleavage expression is a regex, users can add arbitrary cleavage rules, such as methionine cleavage rules.

In the Search step, a user-specified scoring function is applied to a set of queries and a set of candidates for each query. The candidates are a subset of the entire database that matches a user-specified tolerance threshold around the mass of the query precursor neutral mass, as is usually done in other engines to restrict the set of candidates to search against.

The Report step prepares a graphical report containing a mix of performance metrics based on the TDA-FDR methodology, along with score distribution data plotted against

basic PSM statistics, to help identify potential biases or weaknesses in the score function. The Report step also outputs a serialized artifact containing the statistics computed during the report generation for further analysis. This artifact can also be used to generate comparison plots using the `pepid_compare.py` script which is discussed further below.

The Rescoring step applies an arbitrary rescoring function, which is Percolator by default, to the results.

The Final Report step produces the same report artifacts as the Report step, only after the rescoring function is applied.

Pepid phases are reusable and optional throughout its process: unlike other search engines, the intermediary artifacts remain between each step and the software can rerun only the steps of interest to the user. This also enables the user to perform part of the steps, then use custom software to modify the resulting artifact, then proceed with additional steps, without Pepid having to know anything about the intermediary user processing scripts, granting additional pipeline flexibility beyond the basic pipeline proposed by the Pepid system. It also allows users to perform perhaps lengthy processing (especially for peptide generation – that is, in-silico digestion by more complex algorithms and/or spectrum for candidate peptides) only once and to reuse this database for as many query searches as required. Another potential use case is to first generate the preprocessed database on one machine (or multiple machines and then performing a merge step), and making that database available on multiple hosts for search across a cluster (before performing a synchronized final merge of these databases). This design also affords some robustness regarding computational disruptions.

Pepid also provides additional utility scripts: `pepid_compare.py` takes two report artifacts and generates a combined report visualization that makes comparing search results between two conditions convenient. `pepid_files.py` outputs the paths to the files generated by Pepid as part of the search process, given the class of files of interest (for example, just report artifacts or just database artifacts).

2.3.3. Preprocessing

The query preprocessing step simply adds useful metadata, such as the count of peaks in the spectrum and the precursor neutral mass, to the reformated input data, and places it in a database for more efficient further processing. The user can specify criteria used to filter query spectra for quality, such as the peak count range that is allowable and the maximum and minimum precursor mass.

On the other hand, the database processing step is more involved. Its main substeps are as follow: In-silico Digestion, Filtering, Post-translational Modification (PTM) generation, Deduplication.

In the first step, a user-specified regular expression (regex) is applied to the protein sequences in the database to extract "digested" peptides. The regex specified by default corresponds to tryptic digestion using the EXPASY rule. Users can additionally specify maximum missed cleavages. The resulting peptides are then filtered by user-specified minimum and maximum mass and length for the sequences. Next, user-specified maximum variable modifications, variable modification types, and fixed modifications are applied against the peptides kept from the previous step, triggering a refiltering for mass. Finally, duplicate peptide-modification tuples are merged by joining the source protein identifiers to retain a single entry for each unique such tuple.

2.3.4. Search

The search procedure proper first selects a batch of candidates matching a certain tolerance window as specified by the user around the precursor neutral mass of each individual query spectrum, then applies a user-specified scoring function to the batch of candidates for the input query spectrum. Any resulting match scoring more than 0 is kept in the final results database (it is the responsibility of the score implementation to adapt to this 0 cutoff based on other user settings to achieve the desired behavior). The complete artifact takes the form of the query spectrum identifier (the title provided in the input mgf file), the source protein identifier (the protein identification line from the fasta files, or multiple lines merged with semicolons if duplicate peptides were generated in the preprocessing step), the score, and an arbitrary python metadata artifact as a binary blob. It is up to the scoring function implementer to ensure that the artifact contains all the data necessary to apply the rescoring function if such a step is desired later.

The default scoring function provided by Pepid is a combination of an implementation of the Xcorr function based on the Comet codebase and an implementation of the Hyperscore function based on the X!Tandem codebase, a strategy that has been shown in the past to improve search result qualities (Shteynberg et al., 2013). We also modify our Xcorr and Hyperscore implementations in two major ways: first, we expose many implementation details of those algorithms as user modifiable parameters in our configuration file, and second, we expand in multiple directions upon those algorithms. Our extensions are as follows:

— We generalized comet's so-called "flanking peaks" feature to support using arbitrary amount of intensity bins on either side of the current bin during spectrum correction;

— We implement a gaussian kernel and a generalized version of the original comet kernel using the exponential falloff $(\frac{1}{2})^n$ to weigh intensity values n bins away;

— We implement dynamic bin boundary settings based on ppm calculations at the level of individual spectra, supporting 3 operating modes: in "max" mode, the highest m/z in the spectrum is used to compute a m/z difference equivalent and is the basis

for setting the distance between bins; in "precursor" mode, calculation proceeds as in "max" mode, but operates off of the percursor neutral mass; in "bins" mode, bin boundaries are iteratively computed from the maximum allowable mass down to the user-provided minimum mass interval using the distance metric computed as in "max" mode.

We similarly apply modifications to the hyperscore algorithm as follows:

— We keep the top N best-matching sequences, with user-provided N and user-provided series selection criterion (i.e. answering the question "best N what?"): "charge" instructs Pepid to keep the top N charges, counting matches across all series for the same charge level; "series" does the converse (selecting the two best series across all charge levels); "both" selects the best matching vectors (i.e. series-charge sequences).

— We make the theoretical spectrum weighing scheme used by X!Tandem (corresponding to the input parameter 'refine, spectrum synthesis') user-settable and optional.

— We add a score reformulation option that modifies the model to assume that the sum of intensities (all sequences) is caused by the selected best-matching sequence(s) alone (rather than the joint of all selected sequences as in the original X!Tandem model).

We show that some of those modifications can greatly improve peptide identification rates in Section 2.4. Additionally, we deliberately abstain from implementing the X!Tandem hyperscore protein-level peptide score adjustment and the Comet E-Value scoring function because it is well known that both methods defeat appropriate TDA-FDR evaluation (for example, it was noted by Zhang et al. (2012); Gupta et al. (2011b); Matrix Science Ltd (2010); Jeong et al. (2012) and indirectly observed in Elias and Gygi (2009)).

The uncombined version of each scoring function (i.e. Xcorr and Hyperscore) are also made available to users by default by selecting the right function in the configuration file. The base functions were verified to behave identically when using the same parameters compared to their reference implementation in Comet and X!Tandem, respectively.

2.3.5. Input Postprocessing

In Input Postprocessing, a user-specified function receives an input python dictionary-like object describing the corresponding entries in the database that are to be user-processed, and outputs an arbitrary python datum that is then serialized as binary and saved in the database in an extra column. It is up to the user to implement the correct processing functions at the correct stages of the pipeline to make use of this data, or to rely on the primitives already provided by the engine.

In previous work, we added a deep-learning length prediction system as an input post-processing step applied to the set of queries. The length prediction module adds a vector

of probabilities corresponding to the predicted probability of the peptide being of a given length as a metadata item to each query spectrum.

We also add a deep learning-driven spectrum prediction input postprocessing module to the input postprocessing step for candidates. For each candidate, the deep learning module predicts a full spectrum (i.e. both the m/z and the intensity across the entire spectrum, not just the intensity for preselected masses as in Prosit (Gessulat et al., 2019) or DeepMass (Tiwary et al., 2019)), which is added as a metadata item. We do not use this module as the main spectrum generation function because, as with Gessulat et al. (2019); Wilhelm et al. (2021), we find that our predicted spectrum is better used as a feature for a rescoring algorithm like Percolator.

2.3.6. Postsearch

A postprocessing phase following the search, dubbed "postsearch" to avoid ambiguity with other postprocessing phases, is applied before the rescoring phase. When using the length prediction module mentioned above, this phase can add various computed features based on the PSM characteristics to the metadata for each search result. This design was chosen mostly as a demonstration of the pipeline's flexibility, as it could be more efficient to only perform this step as an out-of-pipeline operation as we do to append groundtruth labels to the data (described in implementation), so as to only generate predictions for those entries that will be used for rescoring. Nevertheless, this design allows examining the results to drive potential insights and further peptide identification research.

2.3.7. Rescoring

The rescoring module runs the user-specified function on the input results rows. The default rescoring function generages a Percolator INput file (PIN), and uses Percolator for rescoring, converting the Percolator OUTput file (POUT) at the end of the Percolator rescoring process into a Pepid output tsv. Alternatively, Pepid also provides a random forest-based rescoring method.

We also provide an experimental random forest-based rescorer that is pretrained on Massive-KB (Wang et al., 2018) (a dataset of non-synthetic human peptides with a consensus-based ground truth), in the style of the original formulation of PeptideProphet (Ma et al., 2012). We show that this rescoring method appears to generalize as it successfully rescores both ProteomeTools and One Hour Yeast Proteome search results beyond what Percolator is capable of at both the FDR and FDP level despite no finetuning being applied.

2.3.8. Reports

The Report and Final Report steps are the same, except at different stages of the pipeline: the Final Report operating after rescoring. The result of this step are text file artifacts containing data about PSM quality across FDR thresholds and graphical artifacts summarizing the results (see Figure 4 for an example of the output graphical artifacts).

2.3.9. Out-of-pipeline Operations

The phased, pipeline design of Pepid allows interrupting the process at any time and using arbitrary functions to alter the contents of the databases used to store the queries, candidates or search results. We showcase this capability by implementing such an out-of-pipeline insertion of fields from the mgf file that aren't otherwise recorded by the query processing step (namely the ground truth sequence from ProteomeTools used exclusively for FDP evaluation, which is introduced in the mgf data by the SEQ= field).

2.4. Implementation

Pepid is implemented in Python 3. It is accelerated using a custom multiprocessing module available under `pepid_mp.py` in the source repository, which allows greater flexibility for reporting and debugging than usual python options such as `multiprocessing`. The scoring functions are also accelerated using `numba`, and extensive use of `numpy` and vectorization provides additional speed advantages. For the experimental deep learning facilities, `pytorch` is used for both training and inference.

The `sqlite` database is used as the backing store for the artifacts at each stages. During search, heavy sharding is used to ensure maximum throughput. This means that Pepid is highly sensitive to the storage medium and properties for the destination of the artifacts. To avoid performance degradation due to operations on a large SQLite database, each process creates a new shard whenever a user-specified result count threshold is exceeded.

The condensed output generation (i.e. the output to a human-readable tsv file) and the PIN generation steps can be very slow if operated sequentially, so they are multiprocessed as well. To avoid data races, a file lock is used to negotiate write access to the appropriate file. This is sufficient to speed the process up to 2000% compared to sequential operations.

In the current version, Pepid relies on unix-specific facilities to perform both the multiprocessing (UNIX sockets are used for communication), as well as for the file lock facility.

2.4.1. Search Parameters

Search parameters for the cross-engine comparison using ProteomeTools were selected to match as well as possible between the search engines used, while enabling any special method

available to each engine as would normally be enabled in a real search. A summary in the Pepid configuration format is provided with the software. For the yeast search, parameters for Pepid were optimized so as to represent realistic performance statistics for more accurate comparison on the basis of wallclock time. All parameters used in the experiments in this paper (including mass tolerance, cleavage rule, search algorithm choice and specification, etc.) are available in the pepid repository at `https://github.com/lemieux-lab/pepid` in the configuration files: the `example_proteometools.cfg` file represents the configuration used for the ProteomeTools experiments, the `example_yeast.cfg` file represents the configuration for the one hour yeast proteome experiments, and the `example_proteometools_ml.cfg` file contains the configuration for the ProteomeTools experiment employing all the machine learning tools integrated in pepid.

While Pepid supports rescoring by Percolator or by a partially pretrained random forest method, we only compare engines with percolator due to the pervasiveness of this method across established pipelines. We show the results of the pretrained random forest method compared to previously shown Pepid results to demonstrate generalization and increased identification rates separately. Due to its pretrained nature, this random forest approach cannot equitably be applied to other search engines.

2.4.2. Length Feature Extraction

When the length prediction module is in use, several features are computed in postsearch and added to every PSM from the search step. In particular, we compute the difference between the most-probable predicted length and the candidate length, the predicted probability that the peptide present in the spectrum is of the candidate's length, the difference between the probability of the candidate's length and the best, worst, next best and next worse length probability. Those features are simply extracted during the rescoring step and used as extra features in the Percolator-based pipeline, for example.

2.4.3. Spectrum Generation

We developed a deep learning-based approach to full-spectrum prediction. To the best of our knowledge, the only previous work to have ever successfully used this approach was PredFull (Liu et al., 2020). Our approach differs in several aspects: first, we use fully standard convolutional neural network instead of PredFull's mix of "Spike and Excitation" units and multi-scale convolutions. Second, we output a predicted spectrum for each charge level (equivalent to using 5 different networks that heavily share weights). Third, we constrain the output spectrum so that any intensity less than $1 \cdot 10^{-3}$ is dropped to 0 so as to obtain a sparse spectrum (we find that resulting spectra are usually very sparse, at around 1-3 non-null peaks per 1000 bins, with a total of 50000 bins, i.e. an average of up to 150 peaks

per generated spectrum). We do this mostly due to storage constraints when applying the model during peptide search, although this also acts like a noise removal pass. The model is trained using the ADAM algorithm.

The model's architecture and input processing are presented in Figure 2. It is can be understood as an encoder (a convolutional layer processing the whole sequence at once), 5 processing stages (using convolutional residual layers with a kernel size of 1), and 5 decoders (one per charge level), each independently transmuting a size 1024 latent representation into a size 50000 predicted spectrum corresponding to each input in the sequence, which is then averaged over those sequence elements. The result is a predicted total spectrum in rasterized (i.e. a vector for which each bin has a 0.1 Da width) format for each charge level. This design allows the model to be used efficiently to preprocess a database of candidate peptides, without having to repeat this relatively slow prediction process when iterating upon scoring mechanisms or parameters.

We train the model using the Massive-KB data to avoid potential overfitting vs ProteomeTools as much as possible, and we show some results on ProteomeTools and the Yeast dataset to demonstrate that the resulting model is generalizable. Finally, we apply it during candidate post-processing and store it using the compressed sparse row representation using the SciPy module, version 1.10.0 (Virtanen et al., 2020).

We do not use the same training set as in PredFull because PredFull uses a private subset mix of the ProteomeTools, Massive-KB and Nist data with some data cleaning applied. Instead, we use the Massive-KB data with no ProteomeTools overlap and do not apply special cleaning before processing.

2.4.4. Rescoring

Beside the classical Percolator-based pipeline, we also present a ground truth-based, pretrained, random forest rescorer using the Massive-KB dataset so as to avoid overfitting as much as practical. To demonstrate its capabilities, we measure performance both for ProteomeTools search results with Pepid, and Yeast search results, both on the basis of FDP and FDR identifications over common value ranges. The model is implemented in scikit-learn (Pedregosa et al., 2011) version 1.3.3 as a pipeline composed of a standard scaler followed by the random forest with default parameters (in short, using the gini critierion for impurity and the square root of number of samples for split consideration. Thorough details are available in the software and in pepid's source code). The model is trained in a 10-way cross validation setting and the best-performing model on the basis of separation between ground truth and other hits is saved (along with the scaler). The model and scaler are then loaded and applied to the search results to rescore (i.e. those of ProteomeTools and

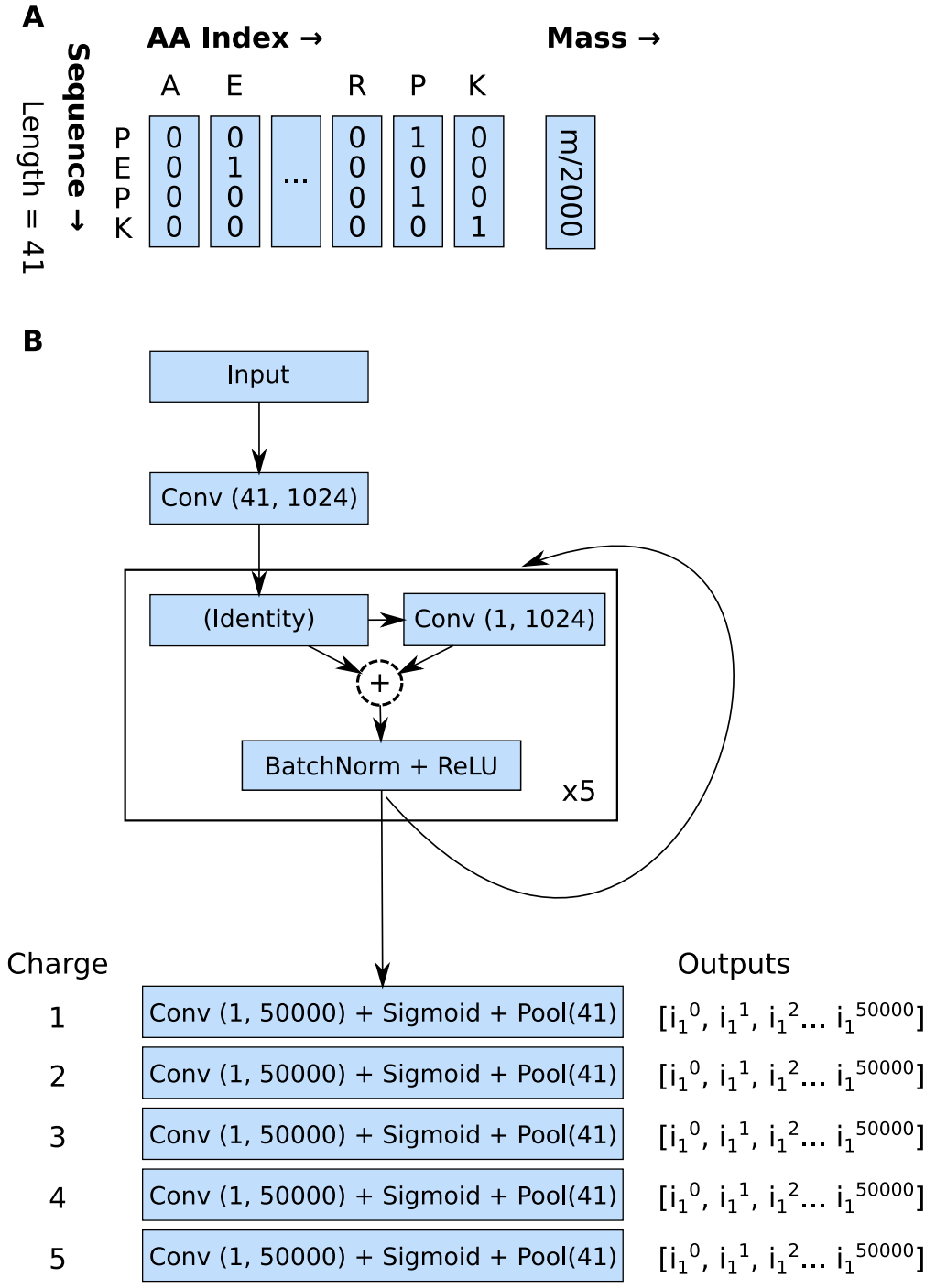


Figure 2 – Architecture and input transformation for the Spectrum Generation model. **A**: The input peptide sequence is encoded similarly to PredFull (Liu et al., 2020) with a one-hot peptide sequence, but without one-hot charge, and using mass divided by a fixed scaling factor (2000 here). **B**: The model architecture is composed of straightforward convolutional layers (kernel size, embedding output size). The final pooling layer averages over the sigmoid outputs across each of the 41 input sequence elements, in effect gathering the consensus from the spectrum implied by each amino acid. The output is 50000 intensity levels for each output charge level. The dashed circle with + represents elementwise addition.

	Massive-KB	ProteomeTools	Yeast
PredFull	0.629	0.573	0.320
Ours	0.632	0.616	0.320

Table 3 – Spectrum generation average cosine similarity compared to PredFull. Performance on Massive-KB is on the held out set only for our method. Note that PredFull uses the pretrained model from predfull.com, which used a combination of proteometools, NIST (Stein, 2008) and Massive-KB data for training.

Yeast in this case), and a second random forest-based rescorer is trained to identify high-scoring vs low-scoring targets (where low-scoring means those with a score below the median score for the collection of PSMs including decoys). This is used as a fine-tuning phase, and the output probability is averaged with the pretrained ground-truth-based random forest probability outputs to generate the final score. The fine-tuning is required to adapt the score to some different experimental conditions, such as search parameters or species under investigation, as we show using the Yeast dataset.

2.4.5. MGF Field Insertion

To compute FDP-based metrics, a separate module (`pepid_mgf_meta.py`) is provided in the Pepid distribution. It takes the configuration file and a field name and inserts the field in each entry of the mgf file into the corresponding entry in the queries database (if the mgf entry was present in the database). This serves also as a demonstrate of working with out-of-pipeline processes to achieve even higher flexibility in overall processing.

2.4.6. Results

Pepid was compared against other search engines to demonstrate that it forms a solid baseline, without which its utility as a research platform would prove limited. It was also compared against itself using different datasets to show how Pepid scales based on dataset size in terms of runtime.

For the comparison between search engines, runtime and identification rates are considered. We use the ProteomeTools dataset as a comparison basis because it contains a large set of clean spectra which, combined with our hardware capabilities, serve to better explore real-world performance metrics than would a smaller dataset.

2.4.7. Performance

Performance results for the spectrum generation task on the whole spectrum are presented in Figure 3. We note that our model is trained only on the Massive-KB data, while the PredFull model is trained on Massive-KB, ProteomeTools, and Nist peptide datasets. Despite this, our model performs slightly better on Massive-KB, and much better on ProteomeTools.

Further comparison to other spectrum generation methods can be found in the PredFull paper (Liu et al., 2020). The indistinguishable, yet low, performance on the One Hour Yeast Proteome dataset is likely due to reaching a ceiling due to the dataset not having a real ground truth (it uses Sequest search results instead) and the spectra in the dataset not being quality-selected, such as by originating from synthetic peptides, or being collected from consensus as in ProteomeTools and Massive-KB respectively. Another potential discrepancy is the difference in NCE. Higher NCE typically results in more relatively pronounced low-mass peaks. Since our model is trained on Massive-KB, it could be underestimating intensities at lower m/z values compared to the expected yeast spectrum.

We note that our method is the only one that can be applied after candidate peptides are generated from in-silico digestion in practice, because our model outputs all possible charged spectra without requiring foreknowledge of the query spectrum's input charge, as is a required feature in all other approaches (thus the need to run the same algorithm once per potential charge state for other approaches, which is often intractable). This kind of design concern is often not apparent when developing tools in isolation rather than as part as a full system like Pepid, therefore further stressing the importance of such tools for peptide identification research.

Runtime performance was evaluated on a machine with the following relevant hardware:

— RAM: 192GB
— CPU: Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz (64 pseudo-cores)
— GPU: 1x Nvidia GeForce RTX 2080 (12GB VRam)
— Disk: Intel Corporation NVMe Datacenter SSD [3DNAND, Beta Rock Controller] Model: SSDPE2KE016T8
— OS: CentOS, kernel version 3.10.0

Figure 1B summarizes the runtime performance of various engines on ProteomeTools so as to demonstrate Pepid viability in a realistic workload, while Figure 3 compares Pepid's scaling performance on our datasets and demonstrates that the added flexibility in the hyperscore and xcorr implementations can be beneficial for identifications in some contexts. The combined xcorr and hyperscore scoring function is used in all experiments. We match our parameters to that of other engines as closely as possible for the cross-engine comparisons. For the Pepid performance scaling experiments, we vary the parameters to show the relative performance of our engine in these scenarios when performing a realistic search function. We find that for both dataset, although especially the yeast dataset, we can improve upon results provided by plain xcorr or hyperscore by using our algorithm extensions, as shown in Figure 3E. We make use of Pepid's phased design to quickly iterate upon search settings or parameters to find the best choices without having to incur input processing and input postprocessing costs (or even search costs when iterating on just pre-rescoring features), while we had to optimize the results the "hard way" for the other search engines, resulting

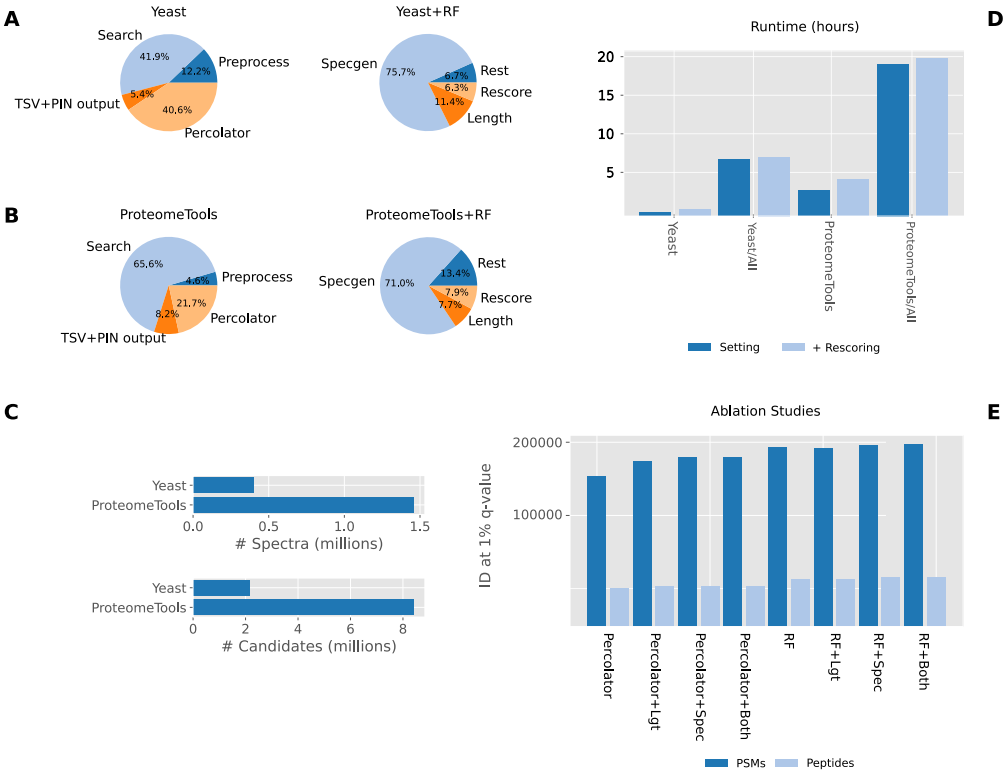


Figure 3 – Performance scaling data for Pepid. Lgt: length predictions. Spec: deep learning spectrum generation. Both: Lgt+Spec. RF: random forest-based rescoring. **A,B**: Runtime breakdown by phase per dataset. Right compares the runtime for computing all machine learning features vs non-ML ("rest") tasks. **C**: Indicative database sizes (queries and candidates). **D**: Overall runtime comparison. **E**: PSM and peptide identifications depending on which of Pepid's modules are in use.

in a slower and more painful iteration process. Due to the relatively painless nature of Pepid development, exposing more parameters and function variants of the hyperscore and xcorr scoring functions was straightforward.

Figure 3A,B demonstrates that while the phased design of Pepid incurs an overhead for preprocessing and output, this overhead diminishes quickly when dataset size increases, allowing search and rescoring wallclock time to dominate the overall process duration. This also appears to be true for the overhead of the machine learning algorithms, although they remain quite expensive to run overall. Thankfully, the phased design employed by Pepid means they only need to be run one per query set or candidate database respectively, and the resulting computed features are "portable" (i.e. they can be reused in different settings without regeneration).

As presented in Figure 1A, despite Pepid keeping all results in the final database on disk, while other engines typically work in-memory and maintain a limited set of top PSMs, it remains competitive in runtime compared to other common search engines. We show that Pepid is faster than currently used search engines, demonstrating that the design's impact

on overall speed is minimal in the current iteration. On the other hand, space usage may be an issue, as storing all the data for the One Hour Yeast Proteome search results takes about 160GB on disk, while for Proteometools, 2.8TB are required. However, deep learning-based methods increase runtime considerably at this time. Though that may be the case, the identification performance impact of our methods are significant in the settings tested in this paper, despite our approaches leaving space for refinement across the board (both for runtime performance and for metric performance).

In Figure 1C, performance between the selected search engines is compared, showing that Pepid performs well compared to other engines in similar parameter settings. A baseline of this level is important as a starting point for further research, as it shows that Pepid works reasonably well without a significant algorithmic update. We believe that this demonstrates once again that Pepid is a suitable tool for further research in peptide identification.

As shown in the figure, with all tools enabled, Pepid outperforms other methods at the FDR level, and greatly outperforms at the peptide level. Moreover, it performs approximately equally to other tools at the FDP level. In a realistic scenario where FDP is not available, it is important for tools to not overestimate FDR, and critical that they do not underestimate it. Other engines include methods that defeat FDR (Gupta et al., 2011a) which are enabled for this comparison. Pepid, on the other hand, is configured in the default files to use conservative parameters, as this is critical to compare algorithmic variations for peptide identification research. Some of these methods, such as Comet's protein-level statistics applied to peptide-level scoring, which effectively perform "local rescoring", cause the "pre-resore" score reported in the figure to be an inaccurate comparison, because in fact, those engines perform partial rescoring or other score adjustments. This is supported by the shrunken identification gap when rescoring is applied across all engines.

2.4.8. Visualization

It is well known that the hyperscore and xcorr scores are biased, notably for peptide length (i.e. they tend to generate higher scores for longer peptides) (Wang et al., 2015; Hubler et al., 2019; Jeong et al., 2012) or charge (Granholm et al., 2012). For peptide identification algorithm research, it is important to have at least two basic metrics: accuracy (that which we aim to optimize, e.g. peptide identification rate) and bias (which we would like to minimize). Pepid currently offers basic visualization aids to expose common sources of biases relating to precursor charge, precursor mass, peptide length, identifications at a selected q-value FDR threshold at the peptide level, spectrum level, and PSM level, and identifiations over FDR, among others. These tools can be used to quickly identify biases, distribution skews, algorithm performance, etc. In Figure 4, a report plot for the xcorr function on the One Hour Yeast Proteome is shown. As previously reported in the literature,

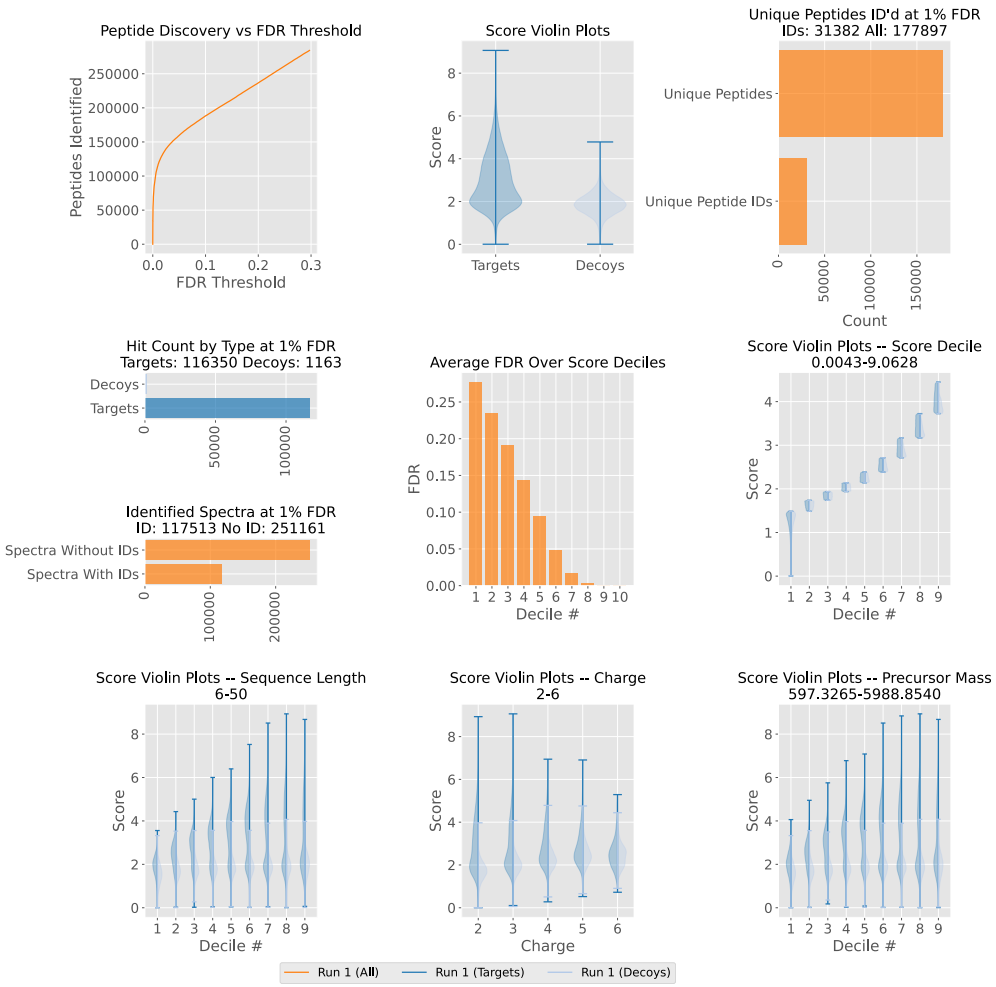


Figure 4 – Example unretouched analysis output from the default Pepid "report" module (run on the One Hour Yeast Proteome data).

we find peptide length bias in the score distribution, demonstrating the potential usefulness of this tool for further development.

2.5. Discussion

We have developed a research platform for peptide identification research that is competitive with state-of-the-art methods in runtime and identification rate performance. Overall, Pepid showcases a design that is specifically oriented toward bioinformatics research and is suitable, in the authors' experience, to combination with deep learning methods, all the while retaining runtime and identification performance in line or superior to commonly used "search-only" engines (that is, engines designed for throughput, not for modification and research).

We have shown that Pepid can easily be adapted to include deep learning methods in the search process, demonstrating the added value of the phased design. The custom

function extension points displayed their value both during the development process (allowing simple configuration modifications to easily compare performance in different conditions) and the research process (allowing the easy extension of the Pepid run to use deep learning models). Furthermore, we have shown that the inclusion of deep learning tools for parts of the peptide identification pipeline can greatly increase identification rates across the board, and that those deep learning tools can be generalized to different species and, to some extent, experimental settings.

Currently, Pepid does not feature a score filtering system. That is because in this initial design, Pepid was developed for peptide identification research first, where we would like to keep as many search results as possible so that we may identify where search algorithms show weaknesses. This data is also usable to train deep learning models in various ways, where the presence of low-quality search results is important to help models explore a richer and more accurate data distribution. In a future update, we plan to address the disadvantage this brings in terms of space constraints.

Pepid's phased design allows it to naturally operate in a distributed environment even though no such feature is officially supported at present; a simple scheme to achieve this is as follows:

— Perform input processing on a single node;
— Migrate the database artifacts to each computer that will be used for search;
— Perform only the search step on each computer, providing a mgf file subset appropriate for each;
— Migrate the resulting search databases to a single node;
— Perform output, report and rescoring on that node, as desired.

In a future update, more attention will be given to user-friendliness, for example by providing easy name association for enzymes instead of requiring the user enter a regular expression directly. In similar veins, An official utility script akin to the comparison and filepath generation scripts described previously will be provided to make other operations, like distributed processing, more convenient. In addition, although Pepid's preprocessing function can be changed by a user's arbitrary function instead, better defaults will be provided, for example expanding modification support and fragmentation types.

Finally, Pepid currently relies on UNIX facilities. While it may or may not operate under Windows using WSL2, a future release may get rid of the UNIX-specific facilitie in favor of a cross-platform option.

Regarding the newly introduced models, the spectrum generator is not aware of NCE during generation and therefore may require postprocessing-based adjustment, or the integration of larger datasets with a range of NCEs, to work better in different NCE scenarios. The lack of quality synthetic or consensus data of sufficient size, especially for non-human

peptides and for non-tryptic digests, makes accurate evaluation of this, and the length prediction model, difficult. Similarly, the lack of reliable ground truth peptide identities makes proper FDP evaluation (to qualify FDR estimation quality) less than ideal, a problem that also affects other search engines as shown in our figures (for example, Comet appears to find fewer peptides than X!Tandem with 1% FDR control, but far more at the 1% FDP level. This shows that X!Tandem's algorithm causes underestimation of FDP by the TDA-FDR method, and that Comet should be preferred in this example. The question to solve is: how can we obtain such confidence when we have no access to ground truth peptides?)

Our search engine is publicly available online at `https://github.com/lemieux-lab/pepid`.

Chapter 3

Mining Mass Spectra for Peptide Facts

Jeremie Zumer, *Institute for Research in Immunology and Cancer*
Sebastien Lemieux, *Institute for Research in Immunology and Cancer, Department of Biochemistry, Université de Montréal, Montréal, Québec H3C 3J7, Canada*

Publication

This article has been submitted to PLOS One and is currently under review

Contributions

Author contributions to this article are as follow:
— Jeremie Zumer performed all other work.
— Sebastien Lemieux advised and corrected the authoring of this paper.

Abstract

The current mainstream software for peptide-centric tandem mass spectrometry data analysis can be categorized as either database-driven, which rely on a library of mass spectra to identify the peptide associated with novel query spectra, or de novo sequencing-based, which aim to find the entire peptide sequence by relying only on the query mass spectrum. While the first paradigm currently produces state-of-the-art results in peptide identification tasks, it does not inherently make use of information present in the query mass spectrum itself to refine identifications, where by "information", we are not referring to the literal peak m/z or intensity value as in classical algorithms like SEQUEST Eng et al. (1994) or even local such patterns as in some PTM localization algorithms such as PTMiner An et al. (2019), but of the complex interaction between the peaks in the whole spectrum which may reveal global clues as to the nature of the peptide. Meanwhile, de novo approaches attempt to solve a complex problem in one go, without any search space constraints in the general case,

leading to comparatively poor results. In this paper, we decompose the de novo problem into putatively easier subproblems, and we show that peptide identification rates of database-driven methods may be improved in terms of peptide identification rate by solving one such subsproblem without requiring a solution for the complete de novo task. We demonstrate this using a de novo peptide length prediction task as the chosen subproblem. As a first prototype, we show that a deep learning-based length prediction model increases peptide identification rates in the ProteomeTools dataset as part of an Pepid-based identification pipeline. Using the predicted information to better rank the candidates, we show that combining ideas from the two paradigms produces clear benefits in this setting. We propose that the next generation of peptide-centric tandem mass spectrometry identification methods should combine elements of these paradigms by mining facts "de novo" about the peptide represented in a spectrum, while simultaneously limiting the search space with a peptide candidates database.
Keywords Deep Learning, Peptide Identification, Proteomics

3.1. Introduction

Mass spectrometry (MS) is currently the only high-throughput method that allows the analysis of peptides and proteins at scale (Duncan et al., 2010). Tandem MS is typically needed to accurately identify the fragments under study as mass conflicts make precursor fragment masses an insufficient proxy for peptide identity in peptide-centric experiments. In this paper, we focus on the tandem MS paradigm and use the term "mass spectrometry" to refer to "data-dependent tandem mass-spectrometry" (in particular in the peptide-centric identification setting) for the sake of brevity. Due to the large and complex data generated by mass spectrometry experiments, analysis by computer software of the mass spectra output is necessary to process the data at a reasonable rate. The main paradigms for such analysis are:

— de novo, or tag-based approaches, where the software tries to identify parts, or complete, sequences from the spectrum (Tran et al., 2017);

— correlation-based approaches (Zhang et al., 2011), which attempt to describe the strength of the correspondence between an experimental spectrum and a spectrum from a database, possibly generated *in silico* based on a candidate sequence (Eng et al., 1994), and

— probabilistic models, which compute the probability that a certain amount of observed peak matches between an experimental spectrum and an expected spectrum (usually generated from a database) occurs by chance as a proxy for the likelihood of a match between the experimental spectrum and the peptide represented by the database spectrum (Cox et al., 2011)

Some of the most widely used mass spectrometry analysis software tools are proprietary. While the high-level ideas behind these methods are sometimes published (Perkins et al., 1999; Cox et al., 2011; Zhang et al., 2011), their finer-grained details remain trade secrets (Ma et al., 2003; Perkins et al., 1999; Cox et al., 2011), which makes iterating or probing these methods complicated. This means that for algorithm development and comparisons, only open-source software is viable (although this is not sufficient, as the design of the tool in question may make the integration of some algorithms hard or impossible). In many contexts, however, closed source tools are preferred for use in practice for peptide identifications (Zhang et al., 2011; Cox et al., 2011). Nevertheless, these proprietary methods see wide use as practitioners empirically find them to work best for their usual workloads, as reported in the literature (Yuan et al., 2014; Välikangas et al., 2017). Many open-source solutions are available (MS-GF+ (Kim and Pevzner, 2014b), Comet (Eng et al., 2012) or Morpheus (Kim and Pevzner, 2014a) for example), but they don't always respond to the community's expectations, often producing results of insufficient quality, although the situation may be changing. For example, IdentiPy (Levitsky et al., 2018) was recently proposed as a new

open-source algorithm that performs better than its other open-source peers, representing a further step toward closing the gap between open-source and proprietary methods.

Even with a wet lab protocol that is optimally tuned for a given (preselected) downstream analysis tool, the results obtained leave space for further improvement on the basis of correct peptide identification rate (Michalski et al., 2011; Zolg et al., 2017; Duncan et al., 2010). In addition, the various state of the art methods share a notable flaw: they rely heavily on input databases (and their specific contents, entry count, composition, and so forth) to query against candidates (Ma et al., 2003; Perkins et al., 1999; Cox et al., 2011; Duncan et al., 2010), which makes results highly dependent on the input database design. In applications where a large specialized or extended database might be useful, the mainstream algorithms place technical restrictions on the size of the databases vs correctly identified peptides, leading the community to develop various sub-optimal tricks to work around these issues (Laumont et al., 2018). Recently, to address the above, the Pepid project (https://github.com/lemieux-lab/pepid) has attempted to build a platform designed specifically for experimenting with peptide search algorithms, which may accelerate improvements in open-source search engines and to the development of more robust identification pipelines.

De novo sequencing methods are tandem mass spectrometry analysis methods that do not make use of databases, instead relying purely on the mass spectrum. Solving de novo sequencing has been attempted for decades with limited success (Lu and Chen, 2003; Chi et al., 2010; Frank and Pevzner, 2005; Cleveland, 2013; Fischer et al., 2005; Tran et al., 2017). However, some of these methods optimize metrics that implicitly imply good performance on other tasks: the recent method DeepNovo (Tran et al., 2017), for example, computes amino acid-level errors over length of ground truth peptide, and errors over length of predicted peptide, in their experiments. Such a metric is consistent with the model's ability to predict peptide length. This suggests that the auxiliary peptide length prediction task may both be easier than, and a required component of, de novo sequencing. Moreover, the fact that DeepNovo performs well for the aforementioned metric, which implicitly measures length prediction performance (despite optimizing for de novo sequencing outcome), strongly suggests that peptide length can be predicted from just the spectrum.

There has been attempts to use de novo sequencing outputs to further improve database searches (Zhang et al., 2011), though those attempts have lead to unconvincing results in the general case, probably in no small part due to the disappointing performance of de novo sequencing methods when used in this context (Lu and Chen, 2003; Chi et al., 2010; Frank and Pevzner, 2005; Cleveland, 2013; Fischer et al., 2005; Tran et al., 2017). On the other hand, de novo sequencing methods have shown successful uses in constrained applications on clean spectra (Bhatia et al., 2012). This demonstrates that de novo methods are sensible as a class of approaches, though hampered by the complexity of spectra from peptides of unconstrained structures. To the best of our knowledge, there has been no attempt so far

to decompose the de novo problem into distinct subproblems (which could be easier than full, end-to-end de novo sequencing, and therefore achieve the subproblem's objective better than de novo approaches can perform end-to-end) and to use solutions to those subproblems for database-driven methods. In this work, we propose one such decomposition. We then focus on peptide length prediction as a subtask of the full de novo sequencing problem to demonstrate its potential usefulness in database-driven peptide identification. Previously, software like MaxQuant (Cox and Mann, 2008) have modeled the relation between peptide length and scores generated by methods such as Mascot (Perkins et al., 1999), and to derive more confident identifications using that model. This is different from our methodology, where we propose to model the length of the peptide based on the spectrum only, and combine that with an external scoring method.

While there can be many ways to consider a decomposition for de novo sequencing, we choose the following model:

(1) Peptide length prediction

(2) Amino acid composition prediction

(3) Amino acid ordering

We show that solving one subproblem from this example decomposition (namely, peptide length prediction) is both reliably achievable using deep learning, as well as proving a valuable asset as part of a state-of-the-art database-driven peptide identification pipeline.

In this paper, we demonstrate that a deep learning-based method can accurately predict peptide length from the spectrum, and that the predictions from such a method can further be used to achieve higher peptide identification rates at a fixed false discovery rate (FDR) as well as at fixed false discovery proportions (FDP) using the SPOT (Frank, 2002) peptide dataset ProteomeTools (Zolg et al., 2017).

3.2. Methods

In order to establish a proper evaluation of the proposed methods, we make use of ProteomeTools (Zolg et al., 2017), a dataset of synthetic (SPOT (Frank, 2002)) peptides. This gives us access to a reasonable ground-truth for the peptide identities in the dataset as well as enabling FDP evaluation rather than relying on target-decoy FDR estimation. The availability of ground-truth data also allows a richer performance comparison between proposed and competing methods. The length model is trained on the Massive-KB (Wang et al., 2018) dataset, so as to keep the evaluation data and the train data as separate as possible to show the applicability of our method. To further demonstrate the generalization power of our length prediction method, we also present measures on the One Hour Yeast Proteome (Hebert et al., 2014) dataset.

We present two measures of the effectiveness of our proposed approach:

(1) Identifications under False Discovery Proportion (FDP)

(2) Identifications under Target-decoy approach-based FDR (TDA-FDR)

Since we have access to reasonable ground-truth, we can compute FDP (sometimes called factual FDR (Jeong et al., 2012)), which is not usually available in non-synthetic datasets where the identity of the peptides in a pool is not known. We compare the relative results (i.e. between the ground-truth and our proposed method) using the aforementioned metrics and note that due to the availability of ground-truth data, the identities at various FDP thresholds is most likely the most relevant. The FDR metrics serve as reference and offer a different, more common view of the results.

3.2.1. Data

ProteomeTools is a dataset of spectra obtained with an Orbitrap Fusion Lumos on a pool of synthetic peptides determined by *in silico* tryptic digestion of the 20 428 sequences of human proteins found in the Swiss-Prot database at the time of recovery of the release in this paper. ProteomeTools data is available on the PRIDE archive with accession ID PXD004732 (Zolg et al., 2017).

In the original ProteomeTools paper, spectrum quality was assessed using the Andromeda score (Cox et al., 2011) for the ground-truth peptide (if it was identified at all by Andromeda). We show statistics related to spectrum quality in Figure 1. The distribution of masses vs peptide length with a linear fit in Figure 1A gives an idea of task difficulty. In Figure 1B, we show the distribution of peptide lengths across the dataset. Figure 1C and D show score distributions, where an Andromeda score of less than 100 is considered "low quality" for the spectrum (Zolg et al., 2017; Cox et al., 2011). It can be seen that while there are many low-quality spectra, the majority are of relatively good quality. We used only the data from the higher-energy collision dissociation (HCD) experiments at a 25 normalized collision energy (NCE) as these seem to produce the best scored results out of the available data according to the analysis in the original ProteomeTools paper (Zolg et al., 2017). The length distribution of peptides in the dataset (fig. 1B) shows a preference for small peptides with 44.28% of the spectra in the dataset covering peptides shorter than 12 amino acids (which corresponds to 38.54% of unique peptide sequences). There are very few samples for large peptides, and the most common peptides are in the length ranges 7-14 (this is the smallest range spanning approximately 2/3 of the data). The full extent of lengths in the dataset is 6 to 40 amino acids.

The One Hour Yeast Proteome is a dataset of non-synthetic yeast peptides with a baseline generated by the Thermo Fisher version of the SEQUEST search engine (Hebert et al., 2014). We use the "batched" dataset, which combines the result of all the hour-long runs. URLs

to the ChorusProject submissions of the data are available in the One Hour Yeast Proteome paper (Hebert et al., 2014).

The Massive-KB dataset builds consensi based on various proteomics experiments submitted to the project, thus combining more realistic peptide pool searches with reasonably confident ground truth identities (Wang et al., 2018). It is the largest of the 3 datasets (about 4x the size of the ProteomeTools' dataset of 63 004 183 spectra) and contains HCD-fragmented spectra of human peptides. The data may be obtained at `https://massive.ucsd.edu/ProteoSAFe/static/massive-kb-libraries.jsp`

To use Massive-KB for evaluation by the length prediction combination with Pepid, a subset of the data where the ground truth peptides may only be modified with fixed modification C(CAM) and variable modification M(ox) was used (by correspondence with ProteomeTools, and due to resource constraints).

In the same setting, the performance for the yeast dataset was obtained by performing the search only on the subset of the data with a ground truth identification (representing about 50% of the dataset) and uses SEQUEST search results rather than known real identities (hence the high accuracy vs pepid).

To ensure that the reported performance metrics generalize to never-before-seen peptides and spectra, the Massive-KB dataset, which is used for training, was split three-way according to the following criteria:

(1) Data (spectra, spectrum metadata, associated peptide) is split into groups depending on the peptide they represent, with each group corresponding to one unique peptide sequence;

(2) The collection of groups is then split 3-ways into sets: 80% for training, 10% for validation, and 10% for testing.

In total, the Massive-KB dataset we used contained 3 178 174 peptides, forming a training set of 2 542 539 peptides, and 317 817 peptides each for testing and validation. All models are trained using the training set, whereas the validation set is used to optimize the model choice and hyperparameters. Finally, the held-out test set is used to report all results in this work for length prediction performance. For experiments combining length prediction with pepid, the Massive-KB, ProteomeTools or One Hour Yeast Proteome dataset are used, as and where indicated, with the ProteomeTools and Yeast data serving as fully held-out data.

3.2.2. Model

We developed a deep learning approach using combined multiple modalities to process the spectrum and metadata, as illustrated in Figure 2A. The mass is encoded as a simple number, while the spectrum is encoding as a rasterized vector of fixed size 50000, summing in each vector element v_i the intensity of all peaks in the spectrum with m/z from $i \cdot 10$ to $(i+1) \cdot 10$.

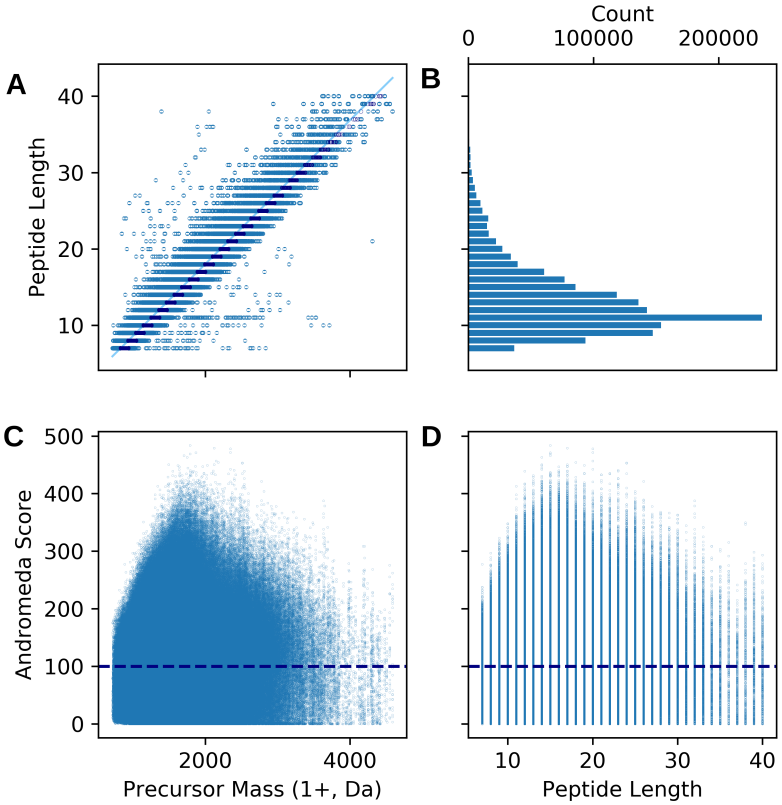


Figure 1 – ProteomeTools dataset properties. **A**: Distribution of peptide lengths vs precursor masses in the dataset. The line represents a linear regression. Dark markers are precursor masses from the spectra with the correct length label using this linear regression, lighter markers are incorrectly labeled. Although the correlation is very high (>0.96), the distribution is wide and thus mass is poorly predictive. **B**: Distribution of peptide lengths in the dataset showing a large proportion of shorter peptides. **C**: Precursor masses vs Andromeda scores. 52.44% of peptide-spectrum pairs pass the suggested quality threshold of 100, as suggested by the Andromeda developers (Cox et al., 2011), shown by the dashed line on the graph. **D**: Distribution of peptide lengths to Andromeda scores.

The model includes a fully connected network, a convolutional neural network, and a different network integrating metadata such as the mass in the captured spectrum (Figure 2B). At this stage, latent representations of those modalities are extracted by each network. We train an auxiliary length prediction task following each of these latent representations (Figure 2D). In Figure 2C, the different networks' outputs are combined using another deep learning network. This network is trained independently of the subnetworks that provide modality-specific latent representations, and gradients do not flow below it: the subnetworks are trained using only the signal from their auxiliary training networks, with a different, independent predictive subnetwork 'head' per modality, as depicted in Figure 2E.

The optimization algorithm is Adam (Kingma and Ba, 2014). When the validation loss does not achieve a new minimum in 10 epochs, the learning rate is divided by 10 and the best performing model is loaded to continue training with this new learning rate. This method is iterated 3 times starting at a learning rate of 10^{-3}, after which no further improvement on the held-out validation set could be observed. The optimization problem is described as follows:

$$\min_{\theta,\phi} \frac{1}{|X|} \sum_{j=1}^{|X|} L(x_{j,\cdot}, y_{j,\cdot})$$

$$L(x_{j,\cdot}, y_{j,\cdot}) = -\sum_{i=1}^{|C|} y_{j,i} \texttt{log}(c_\phi(F_{\bar{\theta}}(x_{j,i}))) - \sum_{n=1}^{N} \sum_{i=1}^{|C|} y_{j,i} \texttt{log}(g_{n,\psi}(f_{n,\theta}(x_{j,i}))$$

where $X \ni (x_i, y_i \in \mathcal{R}^{|C|})$ is the training data, $|X|$ is its size and $|C| = 40 - 6$ length categories are considered (i.e. the model classifies between classes 6 and 40, inclusively), using cross entropy loss. Optimization is performed over hyperparameters ψ for each prediction network used to help drive the subnetworks, θ which are the subnetwork parameters, and ϕ which are the combiner network's hyperparameters. The neural networks f_i are the subnetwork 'processors', F is the concatenation of those networks, N is the count of subnetworks (3 in this case). c is the combiner network, and g_i is the subnetwork prediction output network for subnetwork f_i. Additionally, we use the notation $\bar{\theta}$ to represent θ with gradients disabled, i.e. interpreted as a constant. The full hyperparameters are described in Supplementary Material section B.

The model is designed to predict the probability vector of all classes present in the dataset, thus its output ranges from 6-40 (with index 0 being the probability of length 6, index 1 the probability of length 7, and so forth).

3.2.3. Search Engine Integration

To show that length prediction can improve real peptide search result metrics, we integrate the score from peptide search using Pepid with our length prediction model by producing several features for rescoring by Percolator, as described later in the text. We chose Pepid because its flexible design, good wallclock performance, and high baseline identification performance allowed us to quickly implement the integration and to iterate on feature design for Percolator. This provides a realistic look at what length prediction can achieve in real-world search scenarios.

To showcase the advantage of length prediction in a familiar setting, we compute statistics of the length prediction. We use the following statistics:

(1) The probability output from the model for the candidate's length given the spectrum

(2) The difference between (1) and the best-scoring length

(3) The difference between (1) and the next-best scoring length

(4) The difference between (1) and the previous-best scoring length

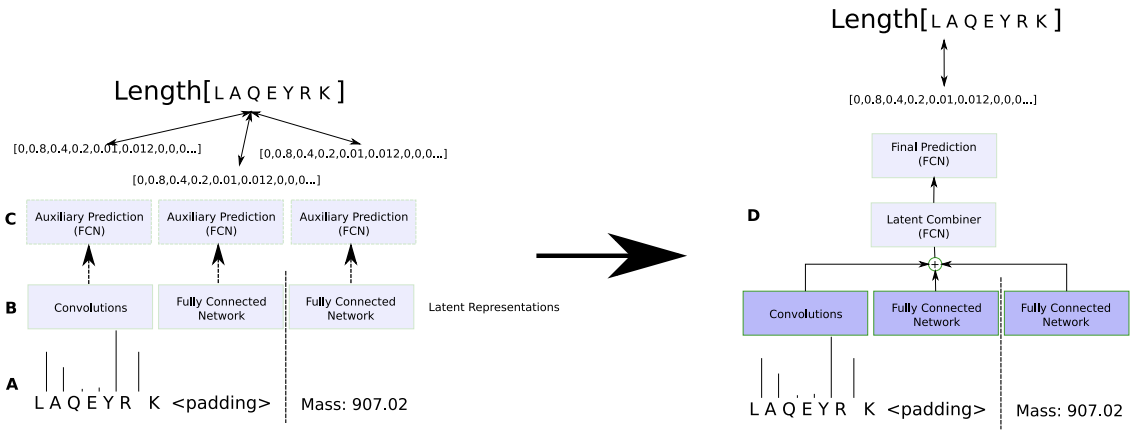


Figure 2 – Diagram representation of the proposed length prediction model, depicting the two-step training procedure. Darker boxes represent layers that don't receive updates from the training process. Step 1, left: **A**: Inputs: fixed-size (i.e. padded) binarized spectrum (for network 1 and 2, independently), and mass scalar (for network 3). **B**: Modality-specific networks process their respective inputs. **C**: Separate prediction heads for each network try to predict length probability vectors. Step 2, right: **D**: The same networks and inputs are used, but the prediction heads are discarded. The latent representations learned in Step 1 are used as input to a new combiner network (taking the concatenated layer outputs as inputs) which is fed into a final prediction network, akin to greedy layerwise pretrianing or other pretraining-based workflows.

(5) The difference between (1) and the worst scoring length

(6) The absolute difference between the best-scoring predicted length and candidate length

(7) The relative difference between the best-scoring predicted length and candidate length (i.e. the absolute difference divided by candidate length)

We provide these statistics to Percolator (Käll et al., 2007) to combine length statistics with the search score, thus providing final identification results.

3.3. Results

We assessed the feasibility of the length prediction task by looking at the distribution of distinct lengths vs the mass of precursors in the dataset, which is shown in Figure 1A, clearly showing that the relation is linear but the range of possible lengths for any mass is wide. To establish a baseline accuracy, we used a simple linear regression, which was trained and tested on the entire dataset (i.e. performance is reported on the same dataset used for training), achieving an accuracy of 38.53%. This suggests the task is not trivial and that methods based on precursor mass and peak-matching-based scores may not be able to exploit implicit peptide length data. We then used a linear regression baseline fit on the Massive-KB data to assess performance (reusing the full set already used for fitting as an indication of performance upper bound), with results as per Table 1. As shown in the table, accuracy and MAE strongly correlate.

	Accuracy (%)	MAE
ProteomeTools	40.8	0.88
Massive	30.4	1.31
Yeast	42.0	0.83

Table 1 – Linear regression baseline performance (trained on Massive-KB, tested on each dataset). MAE: Mean Absolute Error.

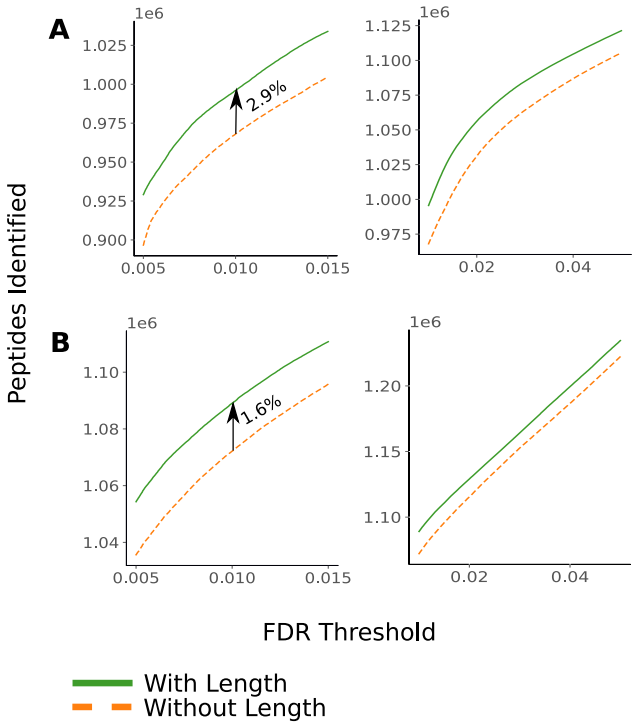


Figure 3 – Pepid with a peptide length "oracle" (values obtained from the ground truth peptides in ProteomeTools). **A**: Identifications across false discovery proportions. **B**: Identifications under target-decoy-based false discovery thresholds.

We note that the performance on the Yeast and ProteomeTools datasets are higher than on Massive-KB, despite the model being trained on the same Massive-KB subset that it is being tested on. This is likely due to Massive-KB being the more realistic of the three, as the ground truth for the Yeast dataset is actually Sequest predictions, while ProteomeTools is a dataset of synthetic peptides.

To demonstrate that length prediction can positively impact peptide identifications beyond state-of-the-art methods, we show in Figure 3 that the Pepid search engine with artificial perfect length predictions generated from the ProteomeTools groundtruth followed by rescoring by Percolator (Käll et al., 2007) can consistently achieve higher identifications across common false discovery ranges. We also show that this holds for FDP (i.e. the real metric of interest).

The results show a modest, but consistent, improvement across the board using just Percolator. Combination using other strategies, such as candidate filtering, using more sophisticated, non-linear rescoring methods, ad hoc scoring functions that taken the length prediction into account, and so

on, could potentially result in further improvements even with the oracle, although it is as yet unclear which of these methods might be most suitable, and how to best combine and prove them for next generation proteomics peptide analysis.

The length prediction model's performance is presented as a confusion matrix in Figure 4. Since this task, to the best of our knowledge, has never been attempted before, we use the Pepid search engine's results as a baseline by taking the length of the best-scoring PSM for each spectrum as a proxy for the "predicted length" from Pepid to be compared to our length prediction model's output. In Table 2, prediction accuracy for the model trained on the Massive-KB dataset is compared based on the range of length of peptide sequences in amino acid, using either the dataset's, or the pepid top-ranking predictions, as ground truth sequences from which the length is computed.

We note that the length prediction performance is worse for longer peptides. This is due in large parts to the lack of representative data in the dataset, as shown in Figure 1. We choose to zoom into the data to the 7-14 length which consists of a much higher amount of samples in the data (approximately 2/3 of the dataset), as it represents the majority of common human peptides (indeed, ProteomeTools peptides are synthetically engineered based on in-silico digests of the human proteome).

We note that the deep learning model works significantly better than the baseline linear regression on Massive-KB (compare Table 1 and Table 2), despite only using a disparate subset when testing the deep learning model, unlike in the linear regression case which presents results obtained on the same set used during fitting, despite showing more modest improvements for ProteomeTools and the Yeast dataset. As noted previously, we attribute these results to the lack of a real ground truth in the yeast dataset, and to the synthetic nature of the ProteomeTools data. Nevertheless, we find that the spectrum data clearly improves length prediction across the board. In particular, we note that the performance between the Pepid top-hit (pepid uses an algorithm very similar to Sequest) for all three predictions are quite closer between the linear regression and the length prediction model, than on the ground truth from the data. Since common search engines provide mass statistics to Percolator for rescoring, this may be a hint that Percolator can effectively exploit mass – but not length – information to achieve improved identification rates.

To further see why the mass-based and spectrum-based length predictions differ, consider the consistent improvements obtained by providing length information to Percolator, both in oracular and in prediction forms. These results demonstrate that (1) real length information (as in the oracle) can help identifications quite a bit despite Percolator's apparent ability to leverage mass to separate hits in rough length-like categories (on account of the inherent correlation between mass and length), and (2) spectrum-based length prediction also provides information in excess of what Percolator can exploit using only its default features and masses.

Additional analysis of the behavior of the length predictions under ProteomeTools are provided in Supplementary Material section B.1. Overall, these results emphasize that data availability has a major impact on performance. Beside this effect, the results also suggest that the length predictor works better on smaller (equivalently: lighter) peptides. The data also shows the strong correspondence between absolute length prediction error and accuracy. Meanwhile, it does not appear to be negatively impacted by PTM counts (so much as data availability) although further

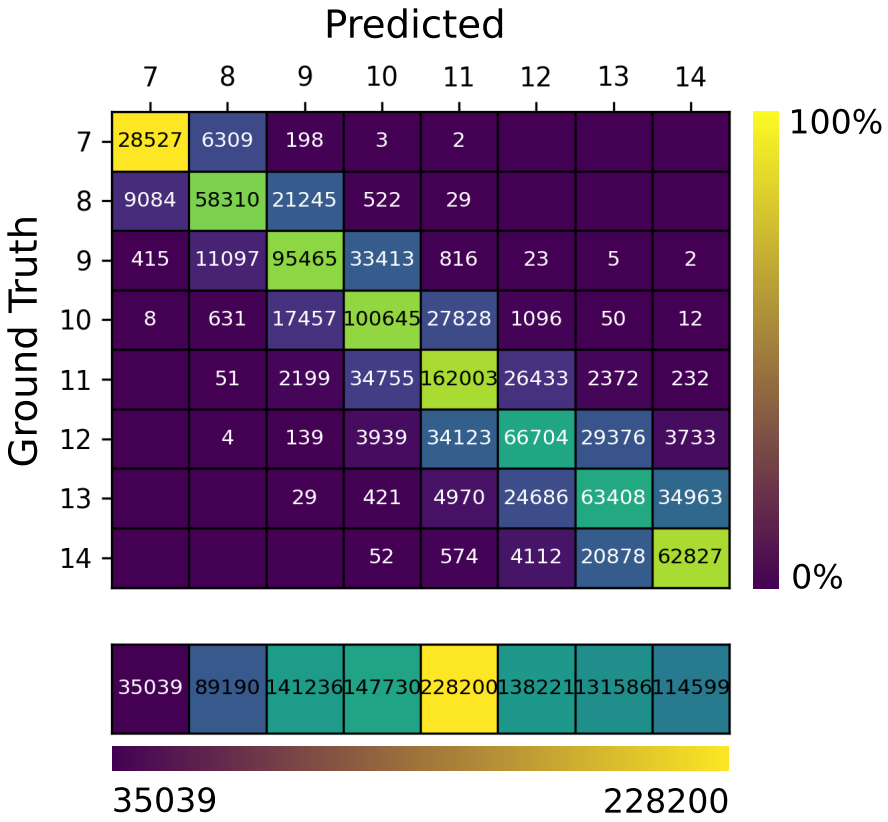


Figure 4 – Length prediction model's confusion matrix showing the total count predicted vs actual ground truth length from the ProteomeTools set. Only lengths 7-14 (corresponding to the most represented classes in ProteomeTools) are shown for easier viewing. Colors in the matrix are for percentage relative to ground truth counts (green is higher, blue is lower). The bottom row displays marginals on all the classes, with coloring as per absolute counts.

Length Range	6-40		7-14	
Ground Truth	Data	Pepid	Data	Pepid
ProteomeTools	42.7	40.4	52.8	50.6
Massive	47.7	35.8	65.1	51.5
Yeast	44.2	43.2	54.2	53.6

Table 2 – Length prediction model's accuracy as a proportion of correctly predicted exact peptide lengths on the test set. **Pepid**: using the Pepid search engine's top-scoring PSM's peptide length as the reference length (i.e. correlation with length distribution of identified peptides). **Data**: using the length of the peptide indicated in the ground truth data (i.e. true peptide)

analysis with large datasets of highly modified peptides would be required to properly assess this. The lack of availability of reliable, large sets of such data limit such analyses at present.

Overall, as shown in Supplementary Material section B, the addition of length prediction (non-oracular) achieves identifications that overlap about 95% of peptides identified without it, but enriches identifications by about 3% while losing less than 2% of peptides identified without it.

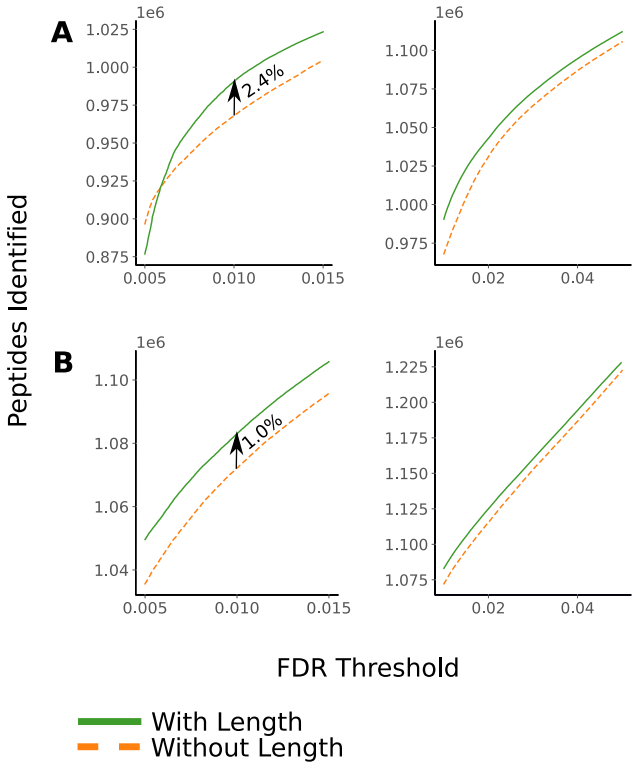


Figure 5 – Pepid with the proposed length prediction model achieves improved performance compared to baseline. **A**: Identifications across false discovery proportions on ProteomeTools. **B**: Identifications under target-decoy-based false discovery thresholds.

Length prediction integration results are shown in Figure 5, demonstrating modest but consistent identification improvements across FDR thresholds similar, but less, than in the oracle version. These results show that the proposed deep learning model can already help improve peptide identification results, despite leaving space for further improvements. This is also reiterated in Table 3 which shows identification numbers under target-decoy (i.e. classic) and ground truth (i.e. FDP based on dataset ground truth peptides) false discovery rates at select common threshold values. In this table, *+ percolator* refers to just the base percolator algorithm, whereas *+ length* also provides the length information to percolator.

	ID @ 1% FDR	ID @ 1% FDP	ID @ 5% FDR	ID @ 5% FDP
Pepid	1 031 333	388 771	1 179 915	1 051 333
+ Percolator	1 072 146	967 607	1 222 694	1 105 596
+ Length	1 083 066	990 546	1 228 584	1 112 063
Oracle	1 089 048	995 584	1 235 215	1 121 710

Table 3 – Performance results for the Pepid platform with and without the length model on ProteomeTools.

We would also like to point out that while the improved identifications over target-decoy based FDR is advantageous, the more important metric is the increased identifications under false discovery proportion. As shown in our results, length prediction is significantly (around 2×) more

impactful when considering true peptide identifications as opposed to target peptide identifications. Due to difficulties in obtaining robust ground truth peptide identities in practice, this metric is often overlooked in favor of the easier to compute TDA-FDR, yet we show here a divergence between the quality of the improvement under both metrics, showing that only considering TDA-FDR during the development and evaluation of refinements to peptide identification pipelines may be misleading. This observation may also impact the feature design for rescoring algorithm like Percolator, or the different score designs for common engines like Comet (Eng et al., 2012) or X!Tandem (Craig and Beavis, 2004): better TDA-FDR identifications may hide lower FDP identifications(Jeong et al., 2012).

3.4. Conclusion

In this work, we have shown that mass spectra contain exploitable information beyond just serving as the basis for matching against a candidate database. We have proposed a deep learning-based method that can exploit query spectra to predict the length of the peptide which generated the fragmentation pattern, and we have demonstrated that this information can successfully be used to improve peptide identification at fixed FDP or FDR thresholds. We believe this is the first time it has been shown that peptide lengths can reliably be recovered directly from the spectrum. Our results were compared to the recent Pepid engine, a modern engine implementing the popular search algorithm Comet's scoring function, demonstrating that length prediction can be of practical interest to improve peptide identification rates in real experiments.

In the typical FDR-based comparison setup, both using FDP and the more classical TDA-FDR estimation, our approach slightly but consistently increased the amount of identified peptides across a range of FDR values, and especially around the common lower FDR ranges (i.e. around 1%). When using FDP, our method similarly enables better separation of false vs true hits, causing an increase in the amount of correctly identified peptides over the baseline Pepid model. Our improvements are consistent across a variety of metrics. For example, length prediction improves unique protein identification at 1% FDR from 61 465 to 62 655 in ProteomeTools, and for unique peptides, the improvement is from 108 161 to 109 794. Thus, at the peptide, PSM, or protein level, and whether comparing FDP or FDR, length prediction consistently improve overall search results. Randomly sampled representative spectra and a venn diagram of unique peptides at 1% FDR are presented in Supplementary Material section B.

The length-based model leaves room for improvement, achieving limited accuracy across the range of peptide lengths present in the ProteomeTools dataset. Despite this low performance on the task the model was trained for, its output can already be used to improve peptide identification, demonstrating that the proposed approach can unlock further improvements for peptide identification. Beside our oracle's performance demonstrating what is possible at the upper bound with a much improved model, this also suggests that extracting more information from the spectrum (rather than trying for full de novo sequencing, for example) may provide more tools to further improve peptide identification in database-driven peptide identification tasks: perhaps properties like hydrophobicity, or the presence of specific amino acids in the sequence, can also be predicted and

used in this manner. Furthermore, these approaches can create a toolbox of algorithms that could lead to similar improvements in a de novo peptide sequencing framework by greatly restricting the potential search space for peptide candidates.

We also note that the performance is much higher in the regime where data is widely available in Massive-KB (the dataset used during training), and suffers primarily when data is not very available. This can best be observed in the full confusion matrix in Supplementary Material section B. That is despite ensuring that the training dataset and the testing datasets (one hour yeast proteome or ProteomeTools) are disjoint.

Combining the length prediction from our model with a classical peaks-matching-based approach (i.e. that of Pepid in our case) is a complex problem. We used a two-step approach in this work and not an end-to-end approach because we aimed to compare the contribution of our de novo subproblem model to an existing baseline following common modern identification workflow practices, and thus to minimize code and architecture changes to just those required to combine our method and the Pepid scoring method. In light of our encouraging results, we believe that an end-to-end scoring approach may further improve peptide identification compared to the results presented in this work. While we do not have the means to combine this method with some of the commonly used software like PeaksDB or MaxQuant's Andromeda as they are proprietary, it would be interesting to verify that this method can improve performance when combined with them.

The results presented in this work are limited in the breadth of parameters in the experiments from which the data was obtained: all the data comes form HCD fragmented spectra at about the same NCE, and two of the datasets were on human peptides with one on yeast peptides. Moreover, our method works well with PTMs, but only static CAM and variable M(ox) were considered. There are several significant hurdles to testing the method on more data spanning a larger set of parameters, most notably the lack of suitably large datasets of that nature with proper ground truth peptide identities. However, our method's ability to generalize to other species and datasets using different instruments, although with similar parameters, suggests that it is not overly reliant on the specific parameters, species and instruments used during training.

Our method only encodes mass in the metadata segment in the current formulation. Previously, we have attempted to encode other metadata, such as the charge, retention time, and precursor intensity, but we have not observed any change in performance when adding these data regardless of format or processing method. We also used different combination methods, including attention-based, multiple metadata elements, and one large metadata input block. It is unlikely that these data would not have any use for length prediction, therefore we avoid presenting these incomplete results. Rather, these results suggest that properly leveraging these data is a more complicated topic that warrants further investigation, and that their proper exploitation require different processing methods.

We believe that the above results clearly show that a mixed approach, i.e. combining ideas from the de novo sequencing and from the database-driven peptide identification, has a lot of potential to improve upon the current state-of-the-art. A database can be used to initially constrain the search space to a manageable subset, and information inherently present in the query spectrum can further be used to reduce this subset by more confidently ranking candidates based on additional

information not present in peak matching alone, although initial attempts at using this method for search space constraints based on retaining peptides based either on a length-based probability threshold or in a window around the most-likely length have been marginal, indicating that performing the task well is more difficult, using this method, than improved identifications with superior scoring. This proposed approach does not suffer from what is arguably de novo sequencing approaches' biggest weakness: the combinatorial search space and increased error probability scaling with peptide length, while making use of their strength: the ability to mine information from the spectrum in advance. Beside peptide length from just the spectrum, data like peptide amino acid composition, or properties like hydrophobicity, could be extracted to further improve identification rates. In addition, predictions from the sequence candidates, such as spectrum prediction or retention time prediction, could further be used in tandem with these spectrum-based feature extractors to improve identifications even further.

Our code is available as open source software as an integrated package in the Pepid framework starting with version v1.1.0. The repository may be accessed at `https://github.com/lemieux-lab/pepid`.

Chapter 4

Targets, Decoys, and Strawmen

Jeremie Zumer, *Institute for Research in Immunology and Cancer*
Sebastien Lemieux, *Institute for Research in Immunology and Cancer, Department of Biochemistry, Université de Montréal, Montréal, Québec H3C 3J7, Canada*

Publication

This article in preparation will be submitted to a journal to be determined at a later date.

Contributions

Author contributions to this article are as follow:
— Jeremie Zumer performed all other work.
— Sebastien Lemieux advised and corrected the authoring of this paper.

Abstract

Many methods have been proposed for evaluating the quality of peptide-spectrum matches (PSMs) identified by peptide search engines. The proteomics community has almost entirely converged upon target decoy-based false discovery rate (FDR) estimation, partly because it is easy to implement and understand, and partly because it stands on solid statistical foundations. However, those foundations rely on assumptions that, when violated, skew, or even completely invalidate, FDR estimation. Currently, the most common identification pipeline setup involves a first search pass using a trusted search engine, and a post-processing pass using so-called rescoring methods. Those methods rely on arbitrary calculated metrics relating to each PSM provided to usually machine-learning based models, that may either be pretrained (Ma et al., 2012) or, more commonly, trained de novo on the provided dataset, usually following a cross-validation protocol in an attempt to avoid overfitting (Käll et al., 2007; Granholm et al., 2012; Degroeve and Martens, 2013). Previous work has shown that after rescoring, the amount of PSMs identified at a selected FDR threshold is approximately the same regardless of search engine used in the first pass (Tu et al., 2015). In this

chapter, we demonstrate that this observation does not hold for the amount of true hits identified by the pipeline and explore other pitfalls of rescoring approaches in a TDA-FDR-evaluated pipeline.
Keywords False Discovery Rate, Target Decoy Approach, Rescoring

4.1. Introduction

Current state-of-the-art peptide search pipelines can be schematically divided into four main phases:

(1) Pre-processing

(2) Candidate generation

(3) Scoring

(4) Post-processing

During pre-processing, some search engines perform quality- and parameter-based filtering of spectra (Craig and Beavis, 2004; Eng et al., 2012; Levitsky et al., 2018): quality-based filtering uses engine-specific metrics and criteria while parameter-based filtering is based on user filtering parameters, such as the number of peaks or their mass. This step also includes "smoothing" the spectra to improve downstream identification rates, e.g. by wavelet methods (Kessner et al., 2008; French et al., 2014) or by retaining only top scoring peaks in a moving window (Levitsky et al., 2018; Craig and Beavis, 2004). Pre-processing can also be an external process: one popular option is MS-Convert and its spectra cleaning tools (Kessner et al., 2008).

In the candidate generation phase, an input database, eventually a protein database, is processed to generate potential peptide candidates for the downstream scoring step. In this step, decoy peptides will also optionally be generated (Eng et al., 2012; Craig and Beavis, 2004; Levitsky et al., 2018). Alternatively, some search engines may also generate decoys for a later so-called "separate search", where decoys and targets do not compete for identification at the scoring step.

Common strategies for candidate generation include the use of sequence libraries (such as from proteomes generated from species-specific genomes Bateman et al. (2020)), using in-silico digestion by trypsin; or the use of spectral libraries, though they are classically limited to known species even as methods exist to artificially generate spectra from sequence shufflings Lam et al. (2009). In the case of sequence libraries, candidates are often generated either by shuffling or reversing the amino acids in the protein before in-silico digestion, or by doing the same to post-digestion peptides. Reversing typically yields slightly less biased sequence distributions Gupta et al. (2011b); Danilova et al. (2019). In this case, spectra are then generated also in-silico based on the sequences obtained. Generally, only m/z values are predicted (based on theoretical fragment masses at given charge levels), although recent work has shown that it is also possible to accurately predict intensity Gessulat et al. (2019); Wilhelm et al. (2021); Tiwary et al. (2019); Liu et al. (2020) (however, perhaps surprisingly, further work is needed to properly use those predicted spectra to improve, rather than decrease, identification rates, with the only known method so far being to provide match scores to rescoring algorithms Wilhelm et al. (2021)). Typically, the main ion series (b- and y-ions) and optionally also common losses (like water and ammonia losses) are modeled (see for example Tabb (2015)), but an overabundance of peaks to match from either side is known to lead to false positive matches with either one high-intensity peaks in the experimental spectrum, or many low-intensity peaks, and to foil some search algorithms (although this is often addressed

via experimental spectrum filtering rather than improved spectrum generation, as in Pirttilä et al. (2022)).

While scoring, candidates from the previously-generated set are chosen and matched against an input spectrum (which will also be referred to in this paper as a "query spectrum"). The scoring process is entirely database-specific (as opposed to previous steps, where most engines have converged to a few or a single operation mode, or rely almost entirely on user-specified parameters). Common archetypes include: probability-like scoring (Perkins et al., 1999) or probability-inspired scoring (Craig and Beavis, 2004; Levitsky et al., 2018), correlation-based scoring (Eng et al., 2012), and peak match counting-based scoring (Craig and Beavis, 2004; Levitsky et al., 2018; Cox et al., 2011). Scoring may be performed using a multi-step approach (especially a two-step approach), as was once common with X!Tandem (Craig and Beavis, 2004), although there is a significant risk of FDR bias when using these approaches (Everett et al., 2010; Bern and Kil, 2011).

Finally, the post-processing step consists mostly of two substeps: score-based filtering (i.e. error control) and rescoring. Score-based filtering removes PSMs that do not match a certain match quality threshold and was often set based on recommendations by the search engine developers and sometimes by experience or examination, or simply by selecting the top N matches for a spectrum (with a user-selected N, such as 1 or 10) (Cox et al., 2011; Eng et al., 2012; Craig and Beavis, 2004). Rescoring is now ubiquitous in peptide searches and is typically performed by external tools such as PeptideProphet (Ma et al., 2012), Percolator (Käll et al., 2007), MS2PIP (Degroeve and Martens, 2013) (which uses Percolator internally), etc.

After a search is complete (i.e. after the post-processing step), search results are filtered based on a quality criterion. The most popular approach is to filter at a select FDR threshold using the hits against the decoy database as a proxy for false hits. Thus, performance metrics reported in the literature usually take the form of PSMs or peptides identified at 1% (or some other level) FDR (Cox et al., 2011; Levitsky et al., 2018; Eng et al., 2012; Craig and Beavis, 2004; Käll et al., 2007; Spivak et al., 2009; Granholm et al., 2012; Degroeve and Martens, 2013; Ma et al., 2012).

In this paper, we explore the validity of the target decoy-based FDR estimation method, especially under the lens of rescoring, which has become ubiquitous to peptide identification workflows. We show, consistent with previous work, that under realistic conditions, the TDA-FDR method misestimates false discovery proportions. We show quantiatively for the first time as far as we're aware that common rescoring methods like Percolator exacerbate this problem and suggest that this is due to the statistics provided by search engines showing different distributions between false hits in the target database and hits in the decoy database, allowing rescoring methods to discriminate more easily and muddying the estimated identifications.

There exists alternative approaches for FDR estimation. One example, the entrapment sequence strategy Granholm et al. (2011); Feng et al. (2017). In this strategy, sequences unlikely to exist in the data of interest is selected (a common choice is a proteome of a different species). Hits to the entrapment database are then assumed to be distributed like false hits to the target database, much like in the TDA-FDR case.

However, such a method has several important downsides: first, it prevents novel peptides similar to those present in the entrapment database from being correctly identified. Second, there is

no evidence that the peptides are distributed similarly in both proteomes, especially when comparing vastly different species like *homo sapiens* vs *archeae*. Third, it suffers from much the same downsides as TDA-FDR, as will be described in more details in the rest of the work.

Several works describe the use of the *Pyrococcus Furiosus* proteome for entrapment in humans. *P. Furiosus* is highly adapted to live in volcanic environments and possesses a very specialized proteome that has virtually no (tryptic) overlap with humans, yet uses the same amino acids in similar abundances with the same kind of protein length distribution. However, if such properties were sufficient, amino acid shuffling-based decoy generation methods would be even more ideal, yet shuffling methods are known to be less reliable than inversion methods (Gupta et al., 2011b; Danilova et al., 2019). As we explore in this paper, such hasty assumptions may be harming FDR estimation and should be re-evaluated.

Another common strategy is that of decoy spectrum libraries such as in Lam et al. (2009), which shuffles the matched peptide sequence amino acids and similarly rearranges the peaks in the spectrum to match the shuffled sequence. However, using spectral libraries also enables the use of decoy-free methods like Shao et al. (2013), which essentially leverages the fact that the refernece spectra in the library are real to derive a distribution of match scores that remains robust outside the dataset used for distribution fitting. As we will show in the rest of the work, though through the lens of TDA-FDR alone, that is likely a more robust strategy in practice as compared to decoy spectra.

Indeed, the majority of this paper applies to any method that assumes and cannot prove at evaluation time some properties of the distribution of a set of peptides that are meant to stand for false target hits, whether by decoy generation, decoy spectral libraries or for entrapment "decoys": all can be strawmen rather than appropriate false hit representations. We focus specifically on the target decoy approach as it has become the most popular on average.

In Tu et al. (2015), as well as in Xu et al. (2013) (among other works), it was observed that following rescoring using Percolator (or Pepid's random forest method), the search engines tested here achieve similar performance on the identified PSM at 1% FDR metric despite more discrepancies before rescoring. In this paper, we show that this does not hold for true hits by using the ProteomeTools dataset (Zolg et al., 2017) of synthetic peptides, which provides clean, high-confidence ground truth for evaluating true peptide identification rates.

Importantly, we also show that TDA-FDR systematically misestimates the false discovery proportion (FDP, also sometimes called factual FDR (Jeong et al., 2012)) in a usual operational scenario, and does so inconsistently depending on search engine used. This suggests that comparison results reported in the literature that do not present FDP performance results and cannot be taken to be valid (that is, they do not show what a reader may be led to believe and likely what the authors believed them to show, as the so-called false discovery rate is actually unrelated to the false discovery proportion across engines). This has previously been reported in the literature (Kandasamy et al., 2009; Jeong et al., 2012; Wang et al., 2008; Bogdanow et al., 2016). We expand on previous work by showing this, and related effects, under application of rescoring by Percolator, rather than arguing using artificial rescoring scenarios.

We also demonstrate that trust in the choice of parameters outputted for the sake of rescoring by modern search engines may be a backdoor to skew reported results regardless of other user-specified settings, and urge practitioners looking for the right search engine for their experiments to re-evaluate their outputs independently using a collection of spectra with confident identification results, rather than trusting FDR-based numbers, let alone identification numbers across search engines.

This paper aims to show that typical TDA-FDR-based evaluation methodologies do not use decoys vs targets for FDR estimation, but rathre targets vs strawmen, defining strawman in line with the Oxford Languages dictionary: a [...] misrepresented [peptide sequence] that is set up because it is easier to [differentiate from targets] than a [database's] real [decoys].

4.2. Data

We use the ProteomeTools (Zolg et al., 2017) data, which is a dataset of Higher-energy Collision Dissociation (HCD)-fragmented synthetic human peptides covering most of the SwissProt human proteome. Proteometools provides an unusually clean dataset, hence an upper bound on the potential amount of identifications at given FDRs, which will serve as our main evaluation metric for demonstration purposes.

ProteomeTools data was acquired from the PRIDE archive with accession ID PXD004732. Due to data corruption in some of the metadata files in the second pool data, we only keep the first pool data. To make comparison between datasets easier by reducing the amount of potential confounding factors, we select only the spectra corresponding to HCD fragmentation at a normalized collision energy of 25, which seems to provide the best overall quality for spectra in the dataset based on the ProteomeTools data analysis presented in Zolg et al. (2017). In total, there are 1 458 831 spectra in the dataset. Some statistics of the data are presented in Figure 1.

For the protein database, the human proteome from SwissProt (Bateman et al., 2020) with ID UP000005640 was used for all experiments. .

4.3. Methods

We use two popular search engines: X!Tandem (Craig and Beavis, 2004), version Alanine, and Comet (Eng et al., 2012), version 2021.02 rev. 0 (commit 3c62af2), which use two different algorithms (X!Tandem uses the Hyperscore, while Comet uses the Xcorr – we do not use the X!Tandem Xcorr plugin). We also use the IdentiPy search engine (Levitsky et al., 2018), version JPR_paper, and the recent Pepid search engine (`https://github.com/lemieux-lab/pepid` v1.2) to show that the effect is not limited to those two popular engines. Search parameters are arranged so that both engines operate with identical parameters wherever possible, and as similar as possible otherwise. For rescoring, the most popular software, which is Percolator (Käll et al., 2007; Spivak et al., 2009; Granholm et al., 2012), version 3.05, is used in all experiments. Execution consisted of the following command in all cases:

```
percolator <file.pin> -Y
```

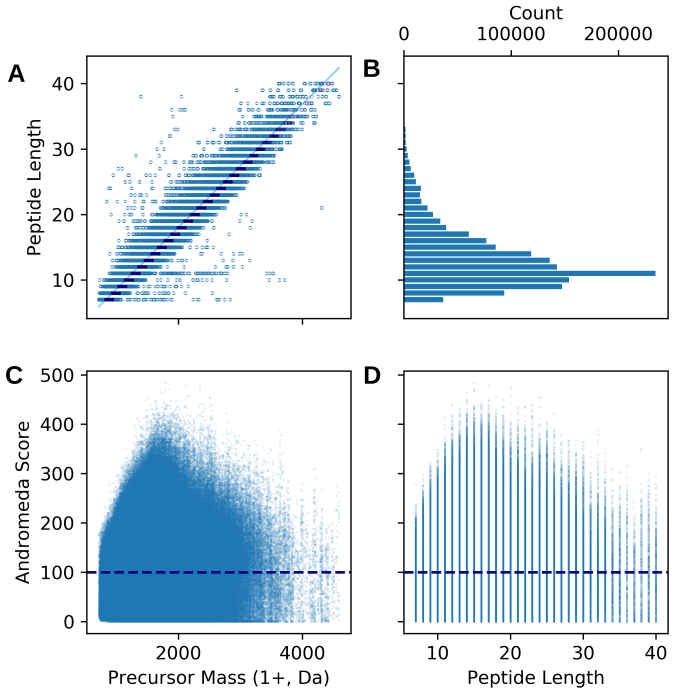


Figure 1 – Statistics of the query spectra from ProteomeTools after filtering as described.

`-m <file_pout_psm.tsv> -M <file_pout_decoys_psm.tsv>`

Threshold calculation is performed using q-value rather than raw FDR (Käll et al., 2007, 2008). This differs only very slightly from FDR in that the minimum FDR (in reverse order) in the score threshold ordering is the current q-value, thus q-value is monotonously increasing whereas FDR is not. This makes calculations more straightforward and rewards search results that rank targets better than decoys overall instead of considering high-ranking decoy runs as being no worse than low-ranking decoy runs in the same overall proportion of decoys to targets. This evaluation metric is computed on the outputted results from each search without relying on search engine filtering or threshold outputs in order to ensure comparability. However, X!Tandem doe not allow outputting more than 1 top-scoring PSM, whereas both Comet and IdentiPy can output arbitrarily many. This has important repercussions for rescoring because Percolator is designed to accept as many search results as is presented for training, and more data typically improves holdout set performance (because the dataset covers a larger portion of the search space) and reduces overfitting rate (because the data is more representative of the true distribution) in machine learning and related algorithms.

Standard searches were run with the engines under consideration and performance was evaluated with and without rescoring with Percolator. The classical PSM at FDR metric with the usual 1% threshold was measured and are shown in Figure 3. Results are similar to those found in the literature (Balgley et al., 2007b; Xu et al., 2013). We note that, owing to the high cleanliness of the data, identification rates at 1% FDR may be higher than in more standard scenarios.

4.4. Results

The goal of rescoring is to help better separate true hits from decoys without enriching for incorrect hits (i.e. targets, excluding true hits). Percolator uses the targets at 1% FDR as a proxy for true hits, and learns to discriminate between members distributed more as per decoy or as per targets at 1% FDR. Hence, disparate distributions of targets at 1% FDR vs true hits would cause percolator to enrich for incorrect hits (if those targets inconsistent with the true hit distribution have higher scores) or true hits (if they tend to display lower scores). The latter is the most desirable scenario. We show what the distribution of scores for these two groups actually look like in Figure 2.

It is reported in the literature that while X!Tandem follows the same pattern as most search engines regarding understimating FDP (i.e. FDP is higher than FDR), Comet, on the other hand, tends to overestimate, or only slightly underestimate, it (Jeong et al., 2012). This is also consistent with search results presented in Figure 3, where X!Tandem seems to perform about as well before rescoring as Comet does after rescoring on the basis of FDR, yet when considering FDP, the reality couldn't be more different: the difference before and after rescoring is significant in every case, not slight as for FDR, X!Tandem perform very poorly even after rescoring compared to Comet, and while Comet seems to underestimate FDP before rescoring, it is much closer after rescoring. In Figure 2, the X!Tandem distributions of interest (i.e. true vs false hits) are far less consistent with the distributions learned by Percolator (i.e. hits passing 1% FDR vs all decoys) than for Comet, which can explain these observations. We also note that the distributions, while much closer in Comet, are still not a direct match.

This example also demonstrates why it is impossible to compare search engines on the basis of their identifications under an FDR threshold: not only are identifications at 1% FDP wildly different than under 1% FDR, indicating a significant misestimation of FDP by the TDA-FDR method, we also arrive at opposite conclusions on the basis of one metric over the other regarding the quantity of error-controlled identifications depending on each search engine. Therefore, identification rates at a fixed FDR do not actually indicate the amount of true error-controlled hits, nor are they even related in a consistent manner.

What happens when a search engine performs a series of post-processing events that amount to rescoring prior to using Percolator? The distributions shown in Figure 2**C,D** show the distribution of Percolator scores after rescoring as an example of the kind of input distributions that Percolator would receive (i.e. repeated Percolator runs). Unlike expectations, the distributions of interest (which separation would be reflected in FDP-based identifications) become more intertwined after the first pass of rescoring, even though the discriminated distributions (which separation impact FDR-based metrics) are more separated. Therefore, search engines that perform rescoring-like tasks (as Comet does by using protein-level metrics, or X!Tandem does in two-phase mode) may be biasing the distribution of scores beyond the kind of skew already implied by the likes of Percolator. (which separation impact FDR-based metrics) are more separated. Therefore, search engines that perform rescoring-like tasks (as Comet does by using protein-level metrics, or X!Tandem does in

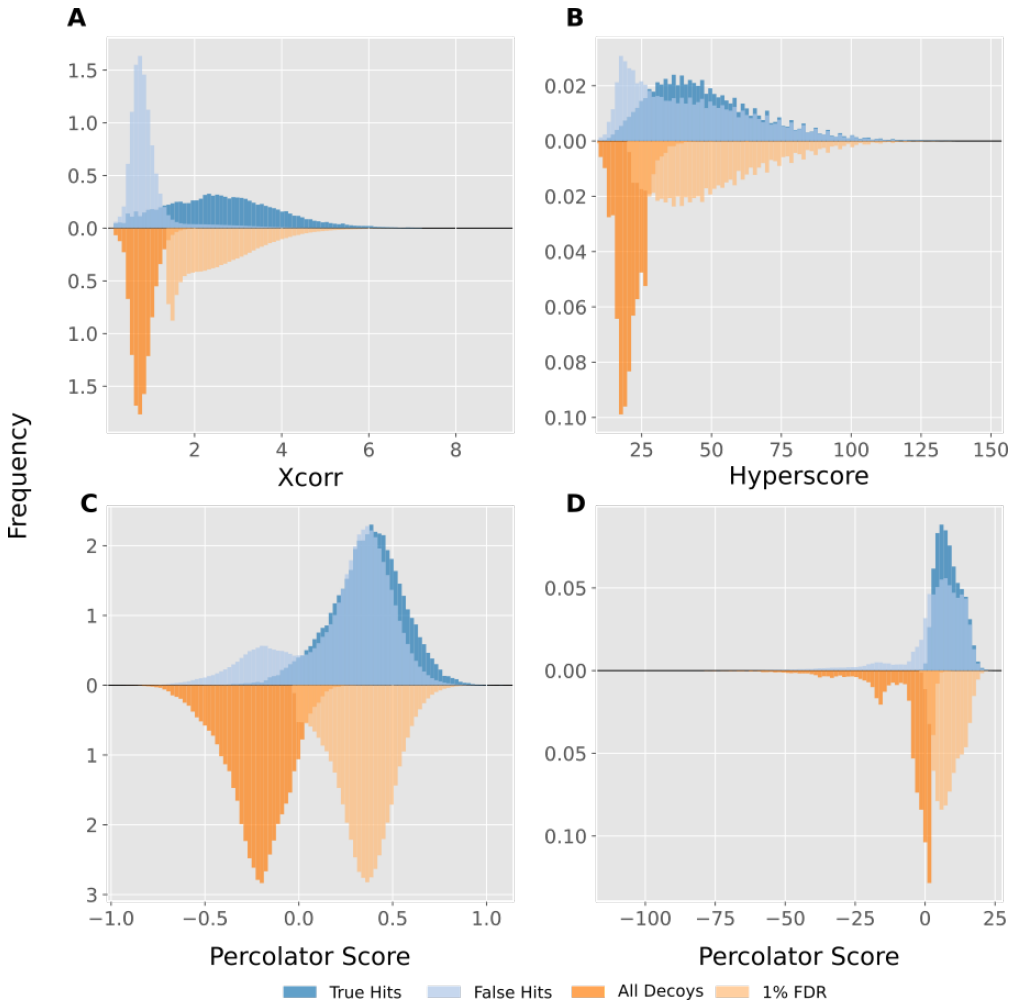


Figure 2 – Distribution of the scores for the subsets of the data of interest for our analysis. The blue distributions, displayed on top of the 0 line, represent the real distributions that we want to separate. The orange distributions, on the bottom, are the distributions that the Percolator algorithm is trained to discriminate. **A,B**: Comet and X!Tandem, respectively, before Percolator. **C,D** Comet and X!Tandem, respectively, after Percolator.

two-phase mode) may be biasing the distribution of scores beyond the kind of skew already implied by the likes of Percolator.

Another useful visualization is the evolution of FDR curves over FDR. We show such curves in Figure 4. Beside the obvious misestimated FDP across the board, we also notice that the misestimation rate is non-linear across the range. This showcases the possibility of a rescoring method (or other score modification algorithm) to, for instance, overwhelmingly underestimate FDR at low FDP values, but misestimating FDR less at higher values, or vice versa. One implication is that even if two engines had similarly good FDP identifications at a fixed level such as 1%, and their results are therefore comparable, this does not necessarily hold at another level, such as 5%, and the adjustment factor would also not necessarily be a simple matter of scaling (rather, the entire FDP misestimation profile would need to be accounted for). The other implication is that

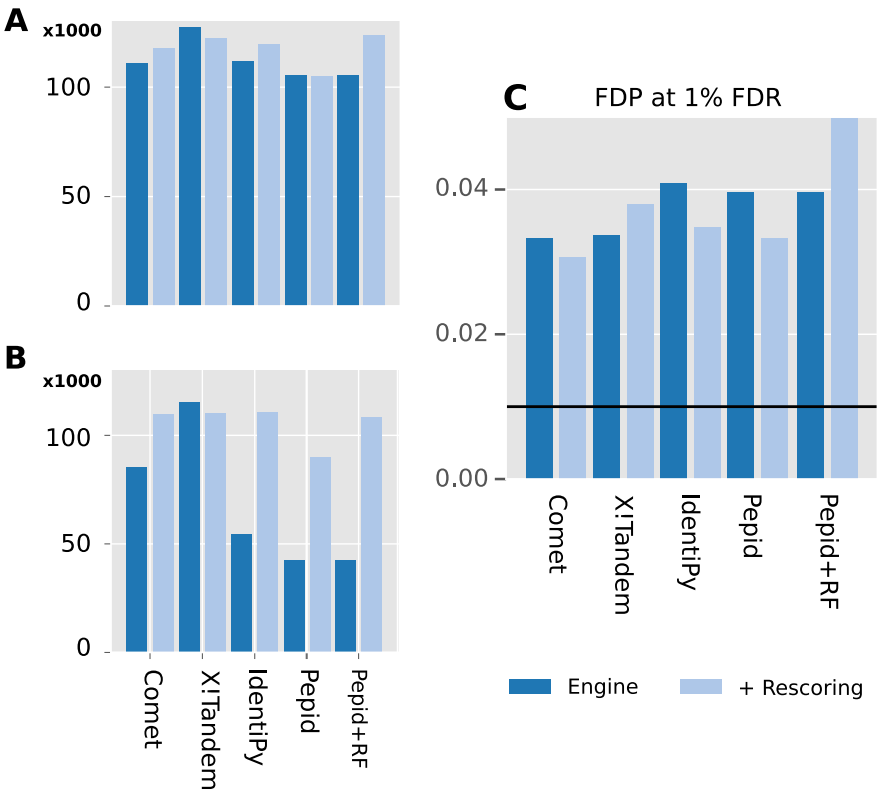


Figure 3 – Identification rates for ProteomeTools data with and without rescoring (Percolator by default, Pepid's Random Forest rescorer in Pepid's case). **A**: TDA-FDR PSM identifications under 1% q-value. **B**: Identifications under 1% q-value using FDP. **C**: Real FDP at 1% FDR.

some systems may be biased in a range of FDR (i.e. there is high uncertainty if FDR is between some bounds, because the FDP values that could have resulted in a given TDA-FDR estimate span a large range) while offering accurate estimation in a different range. This paradigm could encourage the development of different algorithms depending on the FDP control regime of interest for a given experimental setup. Conversely, one might want to choose an algorithm, even knowing it underestimates FDP, based on how linear its misestimation curve is, as this may make identification quality more predictable regardless of experimental concerns.

In both Figure 4 and Figure 3, we show results with and without rescoring. So far we have mostly discussed the overall issues with TDA-FDR in general. However, it is interesting to note just to which extent rescoring affects the FDR vs FDP tableau.

Indeed, in Figure 3**C**, we note that the FDP at 1% FDR between search engines follow very different patterns: Pepid with the random forest rescorer grossly underestimates FDP beyond the level prior to rescoring. Meanwhile, Comet and IdentiPy's overestimation is somewhat subdued after rescoring (as is the case when performing rescoring of Pepid by Percolator). Not so for X!Tandem. We also note by comparing identifications (Figure 3**A,B**) that the amount by which FDP is underestimated does not correspond to the raw amount of identifications. This means detecting these issues in practice ought to be difficult even to experts, using summary identification

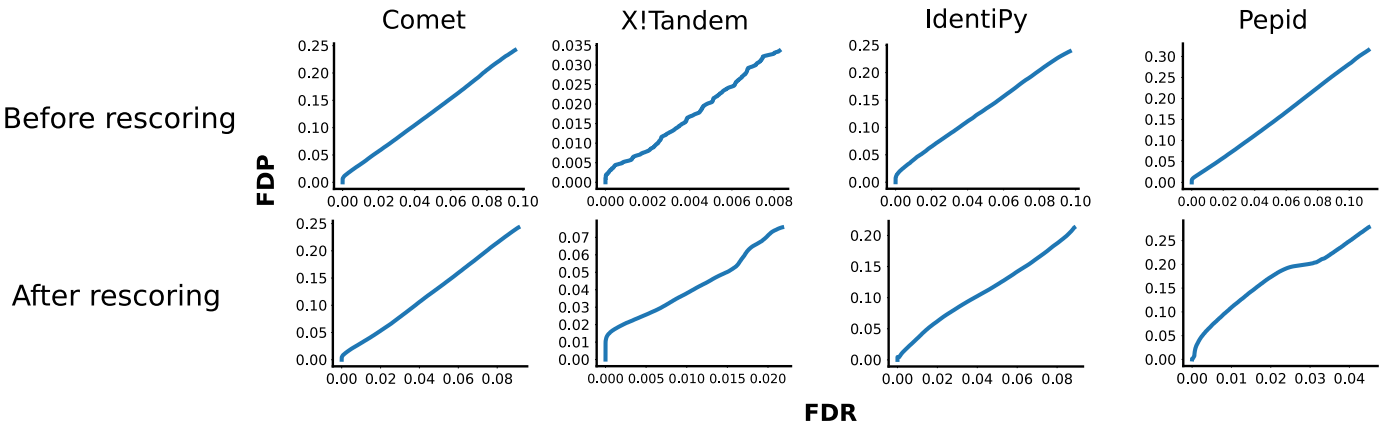


Figure 4 – TDA-estimated FDR curves across the range of FDPs in ProteomeTools.

statistics. In similar veins, we note that prior to rescoring, TDA-FDR-controlled identifications do not necessarily correspond to FDP underestimation. Compare IdentiPy and X!Tandeom.

In Figure 4, two main patterns are readily observable: the overall shape of the curve (i.e. how linear it is) and its inclination (i.e. how misestimation scales with FDP). Regarding the shape of the curve, we notice that on some curves, notably X!Tandem's, rescoring affects an initial overrepresentation "offset value" (i.e. the nearly vertical increase in FDR at the beginning of the curve). Another interesting characteristic is smoothness: compare the unrescored X!Tandem curve with the unrescored Comet curve. Overall, rescoring seems to make the resulting curve less linear (examine before and after curves for Pepid and IdentiPy especially), but has mixed effects on the "offset value" (for example the X!Tandem curve's "offset" is greatly increased, but the IdentiPy "offset" is largely reduced. Likewise, the Comet "offset" benefits from rescoring). Reduced offset values imply less error (as a function of the approximate linear part of the function of misestimation) at lower FDP, but does not necessarily mean the same for higher FDP values (more dependent on the shape of the curve). Smoothness gives clues as to how repeatable identifications can be at any given threshold: smoother curves imply the amount of identifications vary more smoothly with FDP (when threshold is set by TDA-FDR), and is therefore less error prone within a short range of values. Lastly, more linear curves are more predictable regarding overestimation. Here we see that rescoring always makes the curve less linear (while X!Tandem's curve seems smoother, it is now endowed with two (larger) "bumps": one at the beginning of the curve (i.e. the "offset"), and another around 1.7% FDP. The former is smaller in the original curve while the latter is simply not present at all).

Although methods like percolator and the random forest rescorer are non-linear, rendering thorough analysis otherwise difficult, we can examine the separation threshold of various features used as inputs to these tools using Comet and X!Tandem outputs on ProteomeTools as in Figure 5, which illustrate that decoys generated by a state-of-the-art method do not follow the same distribution as false hits as would be required for accurate FDR estimation. This provides hints as to why FDR may be so misestimated: rescoring algorithms can discriminate between decoys and targets on the basis of distribution skews in some of the features rather than through meaningful combination

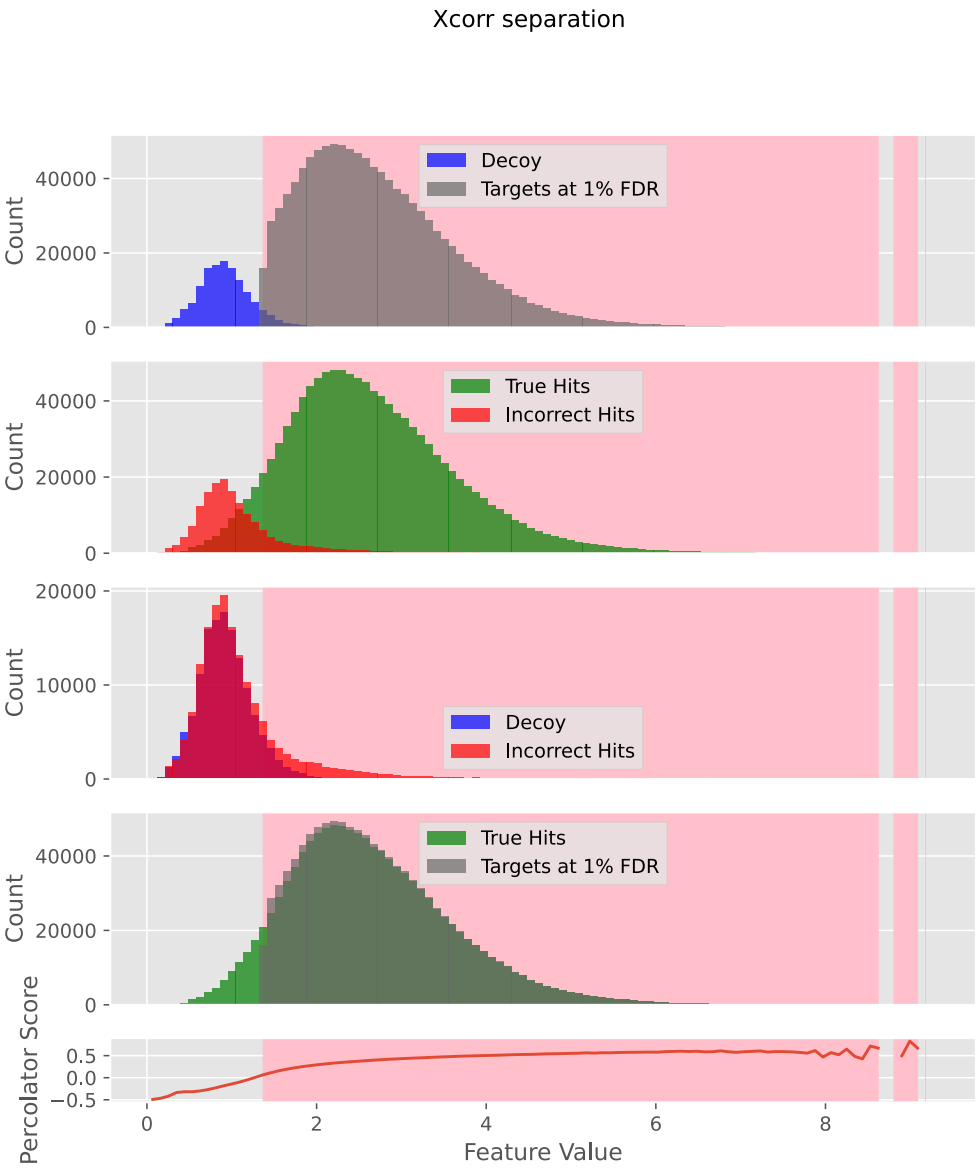


Figure 5 – Distribution skew in ProteomeTools search results using Comet along the Xcorr feature. The distribution pairs match those used by Percolator during model fitting: roughly speaking, Percolator is fit to differentiate between decoys and targets that pass a 1% FDR threshold (top), but the real metric of interest is the separation between true hits and incorrect hits (second from top). There is a slight skew between decoys and incorrect hits (third from top) as well as targets passing 1% FDR threshold and true hits (bottom). The pink zone represents values putatively categorized as true hits by Percolator score at 1% FDR.

of features. Previous work has found that other decoy generation methods suffer from similar or worse problems (see for example Gupta et al. (2011b)). We also show that the same problem can be observed in the Massive-KB dataset Wang et al. (2018) as in Figure 6. Additional results across a slew of parameters from X!Tandem and Comet are presented in Supplemental Material C.1

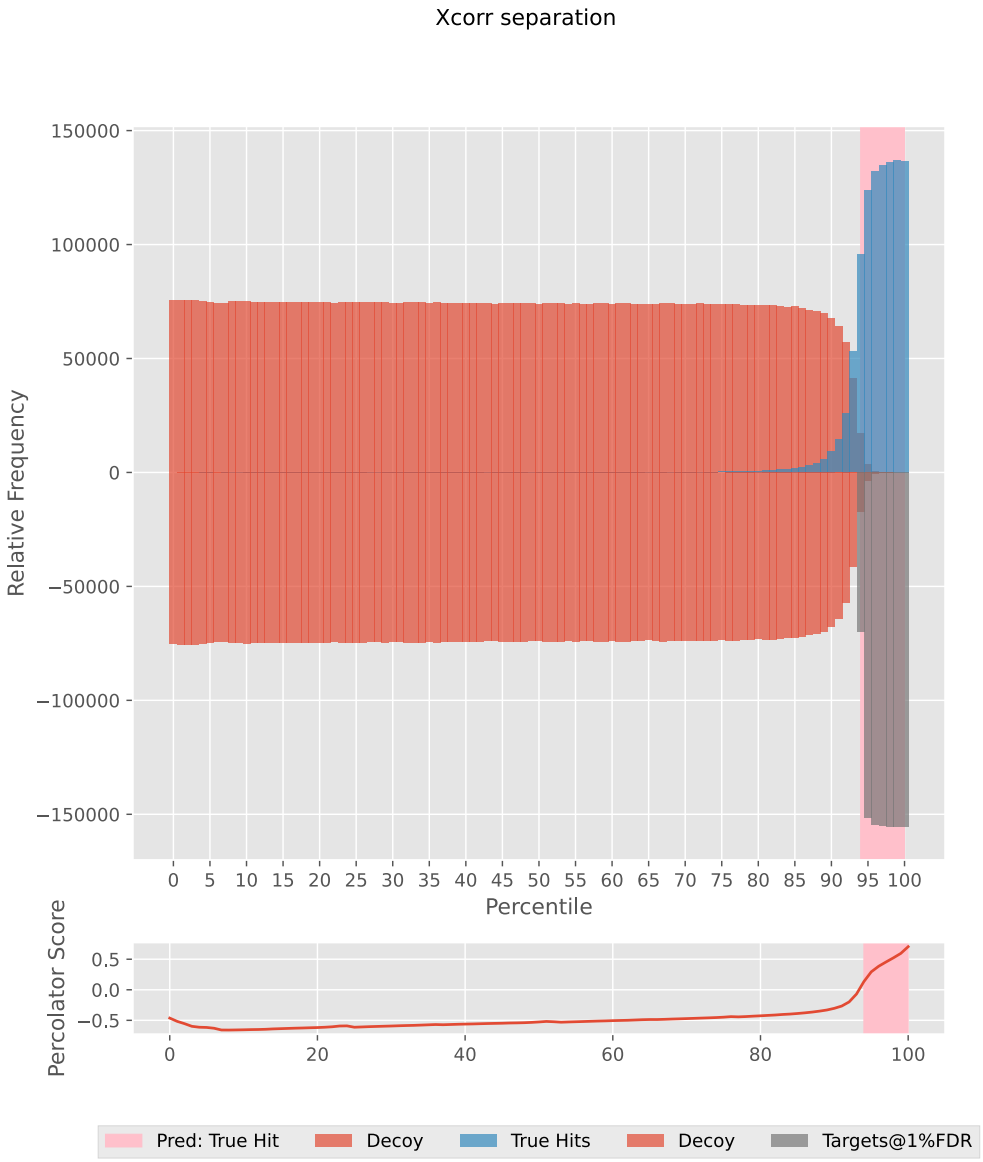


Figure 6 – Distribution skew in Massive-KB search results using Comet along the Xcorr feature with a unifired visualization. The pink zone represents the zone passing a 1% FDR threshold with percolator score for this feature. The red histograms are mirrored for easier comparison in the top and bottom sections. True hits and targets at 1% FDR (resp. blue and grey) are shown to differ, showing distributional bias that could affect TDA-based FDR estimation.

4.5. Discussion

Do so-called Target-Decoy approaches to FDR estimation discriminate between targets and decoys, or between targets and strawmen?

In this paper, we have shown that state-of-the-art evaluation relies on assumptions that can be violated in practice, using realistic data and engines, and that the consequence of violating these assumptions can invalidate search result error correction schemes: FDR estimation no longer knows

about decoys and targets, but rather targets and strawmen. Currently, few datasets with solid ground truth peptide identities are available: some notable ones include ProteomeTools (Zolg et al., 2017), the NIST peptide dataset (Stein, 2008) and MASSive-KB (Wang et al., 2018). These data are biased by the digestion and fragmentation methods and parameters available and, importantly, they are mostly human-only datasets. This greatly limits capacity for accurate evaluation of peptide identifications without relying on hard (or impossible) to verify assumptions about the data, such as using classical FDR metrics, leading to higher risk of development and proliferation of biased methods, as exposed in this paper.

While our results are based on ProteomeTools data mostly, which is a very specific kind of data (synthetic human peptides based on HCD fragmentation and tryptic digestion), our results are consistent with the literature. For example, in Jeong et al. (2012), the authors used ISB Standard Protein Mix database, Arabidopsis and Yeast data with an orbitrap to perform TDA-FDR misestimation analysis. They consistently analyzed several factors that lead to misestimation of FDP by TDA-FDR methods While many of those conditions were artificial and not controllable in practice, they still showcase misestimation issues across different databases, species, search engines, error control formulae, etc.

Our experiments show that current best practices for decoy peptide generation result in distributional differences between decoy and false hit peptides, and moreover that those distribution shifts are very different depending on the engine and dataset used. Previous work like Gupta et al. (2011b); Jeong et al. (2012) show that other methods demonstrate similar issues. Other strategies for decoy peptide generation, such as using deep learning tools, have not been studied substantially. Tools like DeepCoy Imrie et al. (2021), which generate decoy molecules with matched physiochemical properties from an input molecule, may prove to be instrumental in enhancing target-decoy approaches, but to the best of our knowledge, no previous work studies the distributional properties of targets and decoys to assess this. DeepCoy itself is meant for smaller molecules and it is not clear how well it would perform on much larger molecules like peptides. In addition, while this is a good first step, the performance remains less than optimal for this use-case, as even in the molecule case, DeepCoy still only reduces docking-based virtual screening performance as AUC ROC from 0.70 to 0.63.

Overall, we recommend that all peptide identification methods be evaluated on the basis of factual FDR using large datasets with groundtruth peptide identities where possible: for example for any all-purpose search method, or any that are specific to human peptides from tryptic digests. With this baseline established, the reliability of TDA-FDR evaluation on other datasets becomes better understood, which greatly reduces the chance of overfitting this metric, as in our demonstrations.

Additionally, we would like to emphasize that datasets with ground truth peptide identities are not simply useful, but currently a necessity, for accurate evaluation of identification methods, which is a required step in ensuring advancements in peptide identification research instead of "metric hacking". The current lack of coverage for these datasets should be addressed with high priority, perhaps using relatively cheaper methods like that employed by the MASSive-KB project, i.e. based

on consensus from various experiments, when necessary, and preferring synthetic peptide-based data generation, as done by ProteomeTools, when possible.

Chapter 5

Discussion

In this work, we have presented a new platform for peptide identifications. To the best of our knowledge, this is the first platform that is focused on the investigation of novel computational methods for peptide identifications, not just for black-box identification from wetlab experiments. We showed how flexible this method is by providing a suite of functions for various parts of the search process, which can easily be enabled, disabled, or swapped. In addition, our platform allows users to plug in any function in python which they may provide in their own module, so long as it's visible by the `importlib` python library. We further show how the design of this platform benefits research in deep learning tools for peptide identifications by developing a spectrum-based peptide length prediction tool, a full spectrum generator from peptides, and a novel rescoring algorithm based on random forests. We show that the simplicity of integrating these methods in Pepid, and demonstrate that this kind of combination can significantly improve peptide identification rates.

An important feature of Pepid is that it saves all results in a database on disk, which allows further examination of the artifacts from any part of the pipeline. This allows incremental improvements where the pain points lie from quantitative analysis data, a capability never before available. Analysis of the final output of the pipeline (through a large list of hits, not just the top scoring hit as is a common output in many peptide search engines) helped us identify that peptide length prediction could improve identification performance across the board, which we successfully validated (discussed below); what insight is yet to be gained from observing not just the final output artifacts but those of intermediary stages has yet to be explored. For example, perhaps existing correlation-based scoring functions are already performing remarkably well, but that the bottleneck is really theoretical peak sequence generation (evidence for this hypothesis include the significantly improved quality of results when using the same scoring algorithm, but with a library of experimental spectra instead of theoretical sequences). Pepid enables such analysis for the first time.

We develop a deep learning-based length prediction algorithm for peptide sequences from mass spectra. We frame the length prediction problem as a substep of a de novo process, and propose that this problem decomposition approach to de novo peptide sequencing may lead to a collection of easier problems which can be exploited for database-driven peptide identifications, as we have

shown using length prediction, but also potentially to improve de novo peptide sequencing, thus providing a new potential avenue for de novo sequencing development and for research in the combination of de novo and database-driven methods.

As far as we are aware, this is the first time length prediction from mass spectra was demonstrated. We have shown, through the improved rescoring performance of Percolator and our custom rescorer using length prediction, that this feature cannot be extracted by those algorithm from other peptide spectrum match parameters, and we have established through a linear regression baseline based on mass that length prediction from the spectrum is more powerful than from mass alone, indicating that the mass spectrum encodes information that can be used to reconstruct features, like length, of the encoded peptide.

There are two main implications for proteomics and deep learning: first, until now, computational proteomics has been almost chiefly concerned with correlation between experimental and database (theoretical or library-based) data, without regard to what else could be done with information provided by the spectrum. We have shown that peptide length can be retrieved from it and further used to improve peptide identification. This begs the question: what other information can we extract from the spectrum alone in order to improve database-driven searches? De novo practitioners have attempted to predict whole peptide sequences from just the spectrum, with limited successes. Can extracting other data from the spectrum first help drive the de novo process? Of particular interest would be the potential of such approaches to limiting the search space. For example, as we have successfully performed length prediction, this information could be used to limit de novo peptide sequence generation to within a certain peptide length, plus or minus one amino acid (a range within which our prediction model has extremely high accuracy), which may make the de novo process much more likely to yield correct peptides. Second, we have shown that deep learning does, in fact, work to improve peptide identifications in a database-driven setting. Many deep learning algorithms to perform various proteomics tasks have been proposed so far, but few have ever been integrated in search engines. We have shown evidence that this endeavor can be worthwhile. Going forward, deep learning-driven proteomics may become the preferred paradigm for next generation mass spectrometry data analysis.

Finally, we have shown that rescoring methods can mix poorly with target-decoy based false discovery rate estimation, and that ground truth based false discovery should be used when comparing different algorithms, a critical concern during the development and iteration of new scoring algorithms.

Does it matter? These methods have remained in use for years despite their demonstrated flaws. In truth, poorly estimated FDP through the TDA-FDR method does not mean that the entire framework need be eliminated. The hits that pass a so-called 1% FDR threshold are still of relatively good quality and the resulting identification list is still useable. Methods that are demonstratively incompatible with the prerequisite assumptions in TDA-FDR-based FDP estimation do not necessarily skew the results disproportionally. Simply, it is important to understand the existence and scope of this issue because it makes some common use-case of metrics resulting from the use of TDA-FDR, such as comparing identification at a fixed FDR threshold between two conditions or two engines, invalid. This is especially true because derived metrics like identifications

past a fixed FDR change in highly non-linear and hard to predict ways as FDR changes (therefore, as FDR diverges from FDP), so even the scale within which the comparison is invalid cannot easily be assessed, i.e. even if an engine appears to provide 50% more identifications, it may not actually be the case at all.

More concretely, the impacts of the discrepancy between TDA-FDR and FDP in practical settings, especially but not limited to the effect of rescoring, spans various levels in a full mass spectrometry-based proteomics workflow, affecting a variety of domains:

Mass Spectrometry Mass spectrometrists know to visually validate (at least some of) the outputs from identification pipelines. Poorly tuned peptide identification affects all other downstream tasks (from label-free quantification to protein identification) and results in too many false positives and negatives to sort through manually. This may result in mass spectrometrists discarding too many matches out of quality concerns, or to spend too much time on visual inspection compared to what would be required with more robust tuning.

Pipeline Selection In order to minimize experimental expenses, it is important to maximize output quality in the task of interest, so that fewer samples must be processed to achieve the same identification confidence. However, since TDA-FDR is ad-hoc, it is impossible to compare two different engines – core tools of a proteomics pipeline – on even ground, though it remains possible to compare an engine to itself when minor modifications are applied. This means that significant costs may be incurred by poorly tuned pipelines that were selected on the grounds of a highly flawed metric. It should be of interest to the mass spectrometry community to assess those potential costs.

Algorithm Development In the asbsence of robust metrics, it is impossible to develop new algorithms that quantitatively improve identification count or quality. We suggest that using datasets with quality ground truth peptide sequences (like ProteomeTools or Massive-KB) is a reasonable compromise, but those datasets are few and highly biased (for example, both ProteomeTools and Massive-KB are HCD fragmentation datasets of human peptides mostly digested by trypsin). As a result, algorithm development must instead rely heavily on qualitative assessment by expert mass spectrometrists to determine if the algorithm seems to achieve preferred (as opposed to improved) results. This necessarily slows down algorithm development significantly, a problem that some practicioners have decided to avoid by simply not using any robust assessment of their results at all.

In addition, rescoring algorithm are an obvious paradigm for the improvement of search results. They are widely employed in document retrieval domains like semantic text searches and other common applications. However, in the context of peptide search, they are incompatible with the de facto standard error correction scheme in proteomics, that is TDA-FDR, as this metric requires the distribution of false hits and of decoys to match approximately exactly to remain valid. In particular, it is not necessarily clear that TDA-FDR should be assessed after rescoring – that is, it is not clear that rescoring should be seen as a way to improve search counts. Instead, perhaps rescoring should be thought of much like it is in any other domain: as a way, only after an error-controlled search, to fix preference between those items that have already passed our selected threshold, i.e. as a way to learn what to prioritize among the set of hits that pass at our desired fixed FDR level, but not

as a way to further filter results per se. Such an approach would no longer bias the distribution of targets and decoys and allows the separation of concerns between error controlled recall of peptide sequences, and preferential selection for further processing, which can be assessed by different and well understood quality metrics from the document retrieval reranking literature.

Computational Proteomics In practice, it is not clear that the TDA-FDR metric, especially in light of the use of rescoring, is any more or less relevant than the previous paradigm of expert-selected search score threshold. It could be argued that expert selection is more powerful because it essentially allows the mass spectrometrist to insert expert knowledge into the computational pipeline, reducing overall workload. It is not clear that this causes an increase in false positives or negatives after visual filtration of results from either paradigms. On the other hand, the illusion of statistical rigour from common uses of TDA-FDR in combination with rescoring may be damaging across the proteomics pipeline, increasing workload and sample preparation requirements for the same fidelity of results due to the false expectation of achieving a fixed, specified error threshold.

Proteomics is still going through growing pains as the field remains quite young compared to its siblings, genomics and transcriptomics. Many early proteomics tools were plagued with significant algorithmic and mathematical flaws. As the field aged, those flaws have largely been patched or otherwise addressed. Nevertheless, the field has yet to achieve computational maturity and is ripe for improvements in most facets of its computational workflows, in large parts owing to the complicated biochemistry involved, as compared to that of nucleic acids. Deep learning has improved explosively over the past decades and has now become the most powerful approach for processing complicated, noisy data, so long as sufficiently large datasets are available to train neural network models. Since mass spectrometry can produce those large datasets, deep learning seems to be the right fit to push proteomics through to the next level. Indeed, many machine learning and deep learning-based tools have been proposed for proteomics, showing improvements, sometimes by leaps and bounds, on isolated tasks. In this work, we presented new deep learning tools and platforms to further research in both drylab and wetlab proteomics through more robust peptide identifications driven by bioinformatics research and development. Many algorithm types have been proposed in the past, but few have seen any integration in a search pipeline, let alone a full peptide search engine. The proposed Pepid platform forms an ideal testing ground for these methods, to demonstrate that they do, in fact, improve peptide identification in real terms. This forms the basis of the most obvious direction in which this work could be extended. While the deep learning algorithms we developed can be combined with a search engine to yield improved search results in reasonable time, the overhead they incur is still quite large (sometimes increasing total runtime by almost an order of magnitude). Better evaluation-time performance of deep learning models following integration, perhaps through more specialized runtime systems, could further improve the usability of search engines that would heavily rely on deep learning capabilities. A third direction in which this work could be expanded is in attempting to combine multiple parts of the search pipeline using end-to-end deep learning networks, for example by performing direct peptide sequence-spectrum matches instead of generating a theoretical sequence and correlating it to the experimental spectrum. Previous work on the topic includes DeepMatch Schoenholz et al. (2018a)

and SpeCollate Tariq and Saeed (2021). These methods train deep learning models from existing data to directly predict PSM scores. In the case of DeepMatch, the model contains 3 stages: a fragment representation network, followed by a spectral representation network, and a readout network. The fragment representation network maps amino acids to vectors repersenting fragments, using bidirectional LSTMs. The spectral representation network is a fully-connected network that maps the set of fragment reprsentations into a whole-spectrum representation, assigning fragments to m/z bins on the spectrum. Finally, the readout network compares the fragments with corresponding spectrum peaks using the spectral representation mapping. For SpeCollate, the model is composed of two independent paths: one uses bidirectional LSTMs to encode the peptides followed by a fully-connected network to map the output to a latent representation. On the other side, a fully connected network processes the normalized spectrum into a latent representation of the same dimensionality as the first part. Then, the online sextuplet mining method (it is trained on sextuplets of a positive spectrum and peptide pair, a negative example pair for the selected spectrum and a negative example pair for the seleted peptide) is used to optimize a final network which measures match scores between peptides and spectra. However, these methods also do not exploit data that can be extracted from spectra to reduce search spaces and improve PSM scoring, but instead focus on only sequence-to-spectrum matching problem. Moreover, just as it is non-trivial to integrate far better predicted spectra to existing search frameworks, it is unclear how these methods would fare in practice.

Bibliography

Adamo, M. E. and Gerber, S. A. (2016). Tempest: Accelerated MS/MS database search software for heterogeneous computing platforms. *Current Protocols in Bioinformatics*, 55(1).

An, Z., Zhai, L., Ying, W., Qian, X., Gong, F., Tan, M., and Fu, Y. (2019). Ptminer: Localization and quality control of protein modifications detected in an open search and its application to comprehensive post-translational modification characterization in human proteome*. *Molecular & Cellular Proteomics*, 18(2):391–405.

Andreatta, M. and Nielsen, M. (2015). Gapped sequence alignment using artificial neural networks: application to the MHC class i system. *Bioinformatics*, 32(4):511–517.

Arlot, S. and Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4(none).

Balgley, B. M., Laudeman, T., Yang, L., Song, T., and Lee, C. S. (2007a). Comparative evaluation of tandem MS search algorithms using a target-decoy search strategy. *Molecular & Cellular Proteomics*, 6(9):1599–1608.

Balgley, B. M., Laudeman, T., Yang, L., Song, T., and Lee, C. S. (2007b). Comparative evaluation of tandem MS search algorithms using a target-decoy search strategy. *Molecular & Cellular Proteomics*, 6(9):1599–1608.

Bandeira, N. (2010). Protein identification by spectral networks analysis. In *Methods in Molecular Biology*, pages 151–168. Humana Press.

Bateman, A., Martin, M.-J., Orchard, S., Magrane, M., et al. (2020). UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Research*, 49(D1):D480–D489.

Baum, L. E. and Petrie, T. (1966). Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Bern, M. and Kil, Y. J. (2011). Comment on "unbiased statistical analysis for multi-stage proteomic search strategies". *Journal of Proteome Research*, 10(4):2123–2127.

Bhatia, S., Kil, Y. J., Ueberheide, B., Chait, B. T., Tayo, L., Cruz, L., Lu, B., Yates, J. R., and Bern, M. (2012). Constrained de novo sequencing of conotoxins. *Journal of Proteome Research*, 11(8):4191–4200.

Boehm, K. M., Bhinder, B., Raja, V. J., Dephoure, N., and Elemento, O. (2019). Predicting peptide presentation by major histocompatibility complex class i: an improved machine learning approach to the immunopeptidome. *BMC Bioinformatics*, 20(1).

Bogdanow, B., Zauber, H., and Selbach, M. (2016). Systematic errors in peptide and protein identification and quantification by modified peptides. *Molecular & Cellular Proteomics*, 15(8):2791–2801.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152.

Breiman, L. (2001). *Machine Learning*, 45(1):5–32.

Brewis, I. A. and Brennan, P. (2010). Proteomics technologies for the global identification and quantification of proteins. In *Advances in Protein Chemistry and Structural Biology*, pages 1–44. Elsevier.

Bzdok, D., Altman, N., and Krzywinski, M. (2018). Statistics versus machine learning. *Nature Methods*, 15(4):233–234.

Carreira-Perpiñán, M. A. and Hinton, G. (2005). On contrastive divergence learning. In Cowell, R. G. and Ghahramani, Z., editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, volume R5 of *Proceedings of Machine Learning Research*, pages 33–40. PMLR. Reissued by PMLR on 30 March 2021.

Chi, H., Sun, R.-X., Yang, B., Song, C.-Q., Wang, L.-H., Liu, C., Fu, Y., Yuan, Z.-F., Wang, H.-P., He, S.-M., and Dong, M.-Q. (2010). pNovo:de novoPeptide sequencing and identification using HCD spectra. *Journal of Proteome Research*, 9(5):2713–2724.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

CHONG, K. F. and LEONG, H. W. (2012). TUTORIAL ON DE NOVO PEPTIDE SEQUENCING USING MS/MS MASS SPECTROMETRY. *Journal of Bioinformatics and Computational Biology*, 10(06):1231002.

Cleveland, J. P. (2013). Quasinovo: Algorithms for de novo peptide sequencing.

Cleveland, J. P. and Rose, J. R. (2013). Identification of b-/y-ions in MS/MS spectra using a two stage neural network. *Proteome Science*, 11(Suppl 1):S4.

Couté, Y., Bruley, C., and Burger, T. (2020). Beyond target–decoy competition: Stable validation of peptide and protein identifications in mass spectrometry-based discovery proteomics. *Analytical Chemistry*, 92(22):14898–14906.

Cox, J. and Mann, M. (2008). MaxQuant enables high peptide identification rates, individualized p.p.b.-range mass accuracies and proteome-wide protein quantification. *Nature Biotechnology*, 26(12):1367–1372.

Cox, J., Neuhauser, N., Michalski, A., Scheltema, R. A., Olsen, J. V., and Mann, M. (2011). Andromeda: A peptide search engine integrated into the MaxQuant environment. *Journal of Proteome Research*, 10(4):1794–1805.

Craig, R. and Beavis, R. C. (2004). TANDEM: matching proteins with tandem mass spectra. *Bioinformatics*, 20(9):1466–1467.

Danilova, Y., Voronkova, A., Sulimov, P., and Kertész-Farkas, A. (2019). Bias in false discovery rate estimation in mass-spectrometry-based peptide identification. *Journal of Proteome Research*, 18(5):2354–2358.

Degroeve, S. and Martens, L. (2013). MS2pip: a tool for MS/MS peak intensity prediction. *Bioinformatics*, 29(24):3199–3203.

Deutsch, E. W. (2012). File formats commonly used in mass spectrometry proteomics. *Molecular & Cellular Proteomics*, 11(12):1612–1621.

Deutsch, E. W., Perez-Riverol, Y., Chalkley, R. J., Wilhelm, M., Tate, S., Sachsenberg, T., Walzer, M., Käll, L., Delanghe, B., Böcker, S., Schymanski, E. L., Wilmes, P., Dorfer, V., Kuster, B., Volders, P.-J., Jehmlich, N., Vissers, J. P. C., Wolan, D. W., Wang, A. Y., Mendoza, L., Shofstahl, J., Dowsey, A. W., Griss, J., Salek, R. M., Neumann, S., Binz, P.-A., Lam, H., Vizcaíno, J. A., Bandeira, N., and Röst, H. (2018). Expanding the use of spectral libraries in proteomics. *Journal of Proteome Research*, 17(12):4051–4060.

Diedrich, J. K., Pinto, A. F. M., and Yates, J. R. (2013). Energy dependence of hcd on peptide fragmentation: Stepped collisional energy finds the sweet spot. *Journal of the American Society for Mass Spectrometry*, 24(11):1690–1699.

Du, X., Yang, F., Manes, N. P., Stenoien, D. L., Monroe, M. E., Adkins, J. N., States, D. J., Purvine, S. O., David G. Camp, I., and Smith, R. D. (2008). Linear discriminant analysis-based estimation of the false discovery rate for phosphopeptide identifications. *Journal of Proteome Research*, 7(6):2195–2203.

Duncan, M. W., Aebersold, R., and Caprioli, R. M. (2010). The pros and cons of peptide-centric proteomics. *Nature Biotechnology*, 28(7):659–664.

Dupree, E. J., Jayathirtha, M., Yorkey, H., Mihasan, M., Petre, B. A., and Darie, C. C. (2020). A critical review of bottom-up proteomics: The good, the bad, and the future of this field. *Proteomes*, 8(3):14.

Elias, J. E. and Gygi, S. P. (2007). Target-decoy search strategy for increased confidence in large-scale protein identifications by mass spectrometry. *Nature Methods*, 4(3):207–214.

Elias, J. E. and Gygi, S. P. (2009). Target-decoy search strategy for mass spectrometry-based proteomics. In *Methods in Molecular Biology*, pages 55–71. Humana Press.

Eng, J. K., Jahan, T. A., and Hoopmann, M. R. (2012). Comet: An open-source MS/MS sequence database search tool. *PROTEOMICS*, 13(1):22–24.

Eng, J. K., McCormack, A. L., and Yates, J. R. (1994). An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *Journal of the American Society for Mass Spectrometry*, 5(11):976–989.

Everett, L. J., Bierl, C., and Master, S. R. (2010). Unbiased statistical analysis for multi-stage proteomic search strategies. *Journal of Proteome Research*, 9(2):700–707.

Fazal, Z. (2013). Multifactorial understanding of ion abundance in tandem mass spectrometry experiments. *Journal of Proteomics & Bioinformatics*, 06(02).

Feng, X.-d., Li, L.-w., Zhang, J.-h., Zhu, Y.-p., Chang, C., Shu, K.-x., and Ma, J. (2017). Using the entrapment sequence method as a standard to evaluate key steps of proteomics data analysis process. *BMC Genomics*, 18(S2).

Fischer, B., Roth, V., Roos, F., Grossmann, J., Baginsky, S., Widmayer, P., Gruissem, W., and Buhmann, J. M. (2005). NovoHMM: a hidden markov model for de novo peptide sequencing. *Analytical Chemistry*, 77(22):7265–7273.

Frank, A. and Pevzner, P. (2005). PepNovo: de novo peptide sequencing via probabilistic network modeling. *Analytical Chemistry*, 77(4):964–973.

Frank, R. (2002). The SPOT-synthesis technique. *Journal of Immunological Methods*, 267(1):13–26.

French, W. R., Zimmerman, L. J., Schilling, B., Gibson, B. W., Miller, C. A., Townsend, R. R., Sherrod, S. D., Goodwin, C. R., McLean, J. A., and Tabb, D. L. (2014). Wavelet-based peak detection and a new charge inference procedure for MS/MS implemented in ProteoWizard's msConvert. *Journal of Proteome Research*, 14(2):1299–1307.

Frigo, M. and Johnson, S. (2002). The fastest fourier transform in the west.

Gasteiger, E. (2003). ExPASy: the proteomics server for in-depth protein knowledge and analysis. *Nucleic Acids Research*, 31(13):3784–3788.

Geer, L. Y., Markey, S. P., Kowalak, J. A., Wagner, L., Xu, M., Maynard, D. M., Yang, X., Shi, W., and Bryant, S. H. (2004). Open mass spectrometry search algorithm. *Journal of Proteome Research*, 3(5):958–964.

Gessulat, S., Schmidt, T., Zolg, D. P., Samaras, P., Schnatbaum, K., Zerweck, J., Knaute, T., Rechenberger, J., Delanghe, B., Huhmer, A., Reimer, U., Ehrlich, H.-C., Aiche, S., Kuster, B., and Wilhelm, M. (2019). Prosit: proteome-wide prediction of peptide tandem mass spectra by deep learning. *Nature Methods*, 16(6):509–518.

Giese, S. H., Sinn, L. R., Wegner, F., and Rappsilber, J. (2021). Retention time prediction using neural networks increases identifications in crosslinking mass spectrometry. *Nature Communications*, 12(1).

Granholm, V., Noble, W. S., and Käll, L. (2011). On using samples of known protein content to assess the statistical calibration of scores assigned to peptide-spectrum matches in shotgun proteomics. *Journal of Proteome Research*, 10(5):2671–2678.

Granholm, V., Noble, W. S., and Käll, L. (2012). A cross-validation scheme for machine learning algorithms in shotgun proteomics. *BMC Bioinformatics*, 13(S16).

Graves, P. R. and Haystead, T. A. J. (2002). Molecular biologist's guide to proteomics. *Microbiology and Molecular Biology Reviews*, 66(1):39–63.

Gregor, K., Danihelka, I., Graves, A., Rezende, D., and Wierstra, D. (2015). Draw: A recurrent neural network for image generation. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1462–1471, Lille, France. PMLR.

Griss, J. (2016). Spectral library searching in proteomics. *PROTEOMICS*, 16(5):729–740.

Gulcehre, C., Chandar, S., Cho, K., and Bengio, Y. (2017). Dynamic neural turing machine with continuous and discrete addressing schemes.

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., and Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48.

Gupta, N., Bandeira, N., Keich, U., and Pevzner, P. A. (2011a). Target-decoy approach and false discovery rate: When things may go wrong. *Journal of the American Society for Mass Spectrometry*, 22(7):1111–1120.

Gupta, N., Bandeira, N., Keich, U., and Pevzner, P. A. (2011b). Target-decoy approach and false discovery rate: When things may go wrong. *Journal of The American Society for Mass Spectrometry*, 22(7):1111–1120.

Gupta, N. and Pevzner, P. A. (2009). False discovery rates of protein identifications: A strike against the two-peptide rule. *Journal of Proteome Research*, 8(9):4173–4181.

Gupta, N., Tanner, S., Jaitly, N., Adkins, J. N., Lipton, M., Edwards, R., Romine, M., Osterman, A., Bafna, V., Smith, R. D., and Pevzner, P. A. (2007). Whole proteome analysis of post-translational modifications: Applications of mass-spectrometry for proteogenomic annotation. *Genome Research*, 17(9):1362–1377.

Han, X., Aslanian, A., and Yates, J. R. (2008a). Mass spectrometry for proteomics. *Current Opinion in Chemical Biology*, 12(5):483–490.

Han, X., Aslanian, A., and Yates, J. R. (2008b). Mass spectrometry for proteomics. *Current Opinion in Chemical Biology*, 12(5):483–490.

Havilio, M., Haddad, Y., and Smilansky, Z. (2003). Intensity-based statistical scorer for tandem mass spectrometry. *Analytical Chemistry*, 75(3):435–444.

Haykin, S. (1994). *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

He, K., Fu, Y., Zeng, W.-F., Luo, L., Chi, H., Liu, C., Qing, L.-Y., Sun, R.-X., and He, S.-M. (2015). A theoretical foundation of the target-decoy search strategy for false discovery rate control in proteomics. *arXiv preprint arXiv:1501.00537*.

Hebert, A. S., Richards, A. L., Bailey, D. J., Ulbrich, A., Coughlin, E. E., Westphall, M. S., and Coon, J. J. (2014). The one hour yeast proteome. *Molecular & Cellular Proteomics*, 13(1):339–347.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558.

Houel, S., Abernathy, R., Renganathan, K., Meyer-Arendt, K., Ahn, N. G., and Old, W. M. (2010). Quantifying the impact of chimera MS/MS spectra on peptide identification in large-scale proteomics studies. *Journal of Proteome Research*, 9(8):4152–4160.

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

Hubler, S. L., Kumar, P., Mehta, S., Easterly, C., Johnson, J. E., Jagtap, P. D., and Griffin, T. J. (2019). Challenges in peptide-spectrum matching: A robust and reproducible statistical framework for removing low-accuracy, high-scoring hits. *Journal of Proteome Research*, 19(1):161–173.

Imrie, F., Bradley, A. R., and Deane, C. M. (2021). Generating property-matched decoy molecules using deep learning. *Bioinformatics*, 37(15):2134–2141.

Ivakhnenko, A. G. (1971). Polynomial theory of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-1(4):364–378.

Jagannadham, M. V., Gayatri, P., Binny, T. M., Raman, B., Kameshwari, D. B., and Nagaraj, R. (2021). Mass spectral analysis of synthetic peptides: Implications in proteomics. *Journal of Biomolecular Techniques : JBT*, pages jbt.2020–3201–001.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R (Springer Texts in Statistics)*. Springer, hardcover edition.

Jedrychowski, M. P., Huttlin, E. L., Haas, W., Sowa, M. E., Rad, R., and Gygi, S. P. (2011). Evaluation of hcd- and cid-type fragmentation within their respective detection platforms for murine phosphoproteomics. *Molecular & Cellular Proteomics*, 10(12):M111.009910.

Jeong, K., Kim, S., and Bandeira, N. (2012). False discovery rates in spectral identification. *BMC Bioinformatics*, 13(S16).

Journé, A., Rodriguez, H. G., Guo, Q., and Moraitis, T. (2023). Hebbian deep learning without feedback. In *The Eleventh International Conference on Learning Representations*.

Kandasamy, K., Pandey, A., and Molina, H. (2009). Evaluation of several MS/MS search algorithms for analysis of spectra derived from electron transfer dissociation experiments. *Analytical Chemistry*, 81(17):7170–7180.

Kessner, D., Chambers, M., Burke, R., Agus, D., and Mallick, P. (2008). ProteoWizard: open source software for rapid proteomics tools development. *Bioinformatics*, 24(21):2534–2536.

Khirirat, S., Feyzmahdavian, H. R., and Johansson, M. (2017). Mini-batch gradient descent: Faster convergence under data sparsity. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2880–2887.

Kiefer, J. and Wolfowitz, J. (1952). Stochastic Estimation of the Maximum of a Regression Function. *The Annals of Mathematical Statistics*, 23(3):462 – 466.

Kilpatrick, L. E. and Kilpatrick, E. L. (2017). Optimizing high-resolution mass spectrometry for the identification of low-abundance post-translational modifications of intact proteins. *Journal of Proteome Research*, 16(9):3255–3265.

Kim, M., Zhong, J., and Pandey, A. (2016). Common errors in mass spectrometry-based analysis of post-translational modifications. *PROTEOMICS*, 16(5):700–714.

Kim, S. and Pevzner, P. A. (2014a). MS-GF+ makes progress towards a universal database search tool for proteomics. *Nature Communications*, 5(1).

Kim, S. and Pevzner, P. A. (2014b). MS-GF+ makes progress towards a universal database search tool for proteomics. *Nature Communications*, 5(1).

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980.

Kitamura, N. and Galligan, J. J. (2023). A global view of the human post-translational modification landscape. *Biochem. J.*, 480(16):1241–1265.

Kitata, R. B., Yang, J.-C., and Chen, Y.-J. (2022). Advances in data-independent acquisition mass spectrometry towards comprehensive digital proteome landscape. *Mass Spectrometry Reviews*.

Kleinberg, B., Li, Y., and Yuan, Y. (2018). An alternative view: When does SGD escape local minima? In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2698–2707. PMLR.

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

Käll, L., Canterbury, J. D., Weston, J., Noble, W. S., and MacCoss, M. J. (2007). Semi-supervised learning for peptide identification from shotgun proteomics datasets. *Nature Methods*, 4(11):923–925.

Käll, L., Storey, J. D., MacCoss, M. J., and Noble, W. S. (2008). Assigning significance to peptides identified by tandem mass spectrometry using decoy databases. *Journal of Proteome Research*, 7(1):29–34.

Lam, H., Deutsch, E. W., and Aebersold, R. (2009). Artificial decoy spectral libraries for false discovery rate estimation in spectral library searching in proteomics. *Journal of Proteome Research*, 9(1):605–610.

Lam, H., Deutsch, E. W., Eddes, J. S., Eng, J. K., King, N., Stein, S. E., and Aebersold, R. (2007). Development and validation of a spectral library searching method for peptide identification from MS/MS. *PROTEOMICS*, 7(5):655–667.

Lam, H., Deutsch, E. W., Eddes, J. S., Eng, J. K., Stein, S. E., and Aebersold, R. (2008). Building consensus spectral libraries for peptide identification in proteomics. *Nature Methods*, 5(10):873–875.

Larsen, J. and Goutte, C. (1999). On optimal data split for generalization estimation and model selection. In *Neural Networks for Signal Processing IX: Proceedings of the 1999 IEEE Signal Processing Society Workshop (Cat. No.98TH8468)*, pages 225–234.

Laumont, C. M., Daouda, T., Laverdure, J.-P., Bonneil, É., Caron-Lizotte, O., Hardy, M.-P., Granados, D. P., Durette, C., Lemieux, S., Thibault, P., and Perreault, C. (2016). Global proteogenomic analysis of human MHC class i-associated peptides derived from non-canonical reading frames. *Nature Communications*, 7(1).

Laumont, C. M., Vincent, K., Hesnard, L., Audemard, É., Bonneil, É., Laverdure, J.-P., Gendron, P., Courcelles, M., Hardy, M.-P., Côté, C., Durette, C., St-Pierre, C., Benhammadi, M., Lanoix, J., Vobecky, S., Haddad, E., Lemieux, S., Thibault, P., and Perreault, C. (2018). Noncoding regions are the main source of targetable tumor-specific antigens. *Science Translational Medicine*, 10(470).

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Hubbard, W., and Jackel, L. (1989). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, S., Park, H., and Kim, H. (2022). False discovery rate estimation using candidate peptides for each spectrum. *BMC Bioinformatics*, 23(1).

Levitsky, L. I., Ivanov, M. V., Lobas, A. A., Bubis, J. A., Tarasova, I. A., Solovyeva, E. M., Pridatchenko, M. L., and Gorshkov, M. V. (2018). IdentiPy: An extensible search engine for protein identification in shotgun proteomics. *Journal of Proteome Research*, 17(7):2249–2255.

Li, J., Smith, L. S., and Zhu, H.-J. (2021). Data-independent acquisition (DIA): An emerging proteomics technology for analysis of drug-metabolizing enzymes and transporters. *Drug Discovery Today: Technologies*, 39:49–56.

Li, Y., Cai, T., Zhang, Y., Chen, D., and Dey, D. (2023). What makes convolutional models great on long sequence modeling? In *The Eleventh International Conference on Learning Representations*.

Lin, S. M., Zhu, L., Winter, A. Q., Sasinowski, M., and Kibbe, W. A. (2005). What is mzxml good for? *Expert Review of Proteomics*, 2(6):839–845.

Liu, K., Li, S., Wang, L., Ye, Y., and Tang, H. (2020). Full-spectrum prediction of peptides tandem mass spectra using deep neural network. *Analytical Chemistry*, 92(6):4275–4283.

Lu, B. and Chen, T. (2003). A suboptimal algorithm for de novo peptide sequencing via tandem mass spectrometry. *Journal of Computational Biology*, 10(1):1–12.

Ma, B., Zhang, K., Hendrie, C., Liang, C., Li, M., Doherty-Kirby, A., and Lajoie, G. (2003). PEAKS: powerful software for peptide de novo sequencing by tandem mass spectrometry. *Rapid Communications in Mass Spectrometry*, 17(20):2337–2342.

Ma, C. W. M. and Lam, H. (2014). Hunting for unexpected post-translational modifications by spectral library searching with tier-wise scoring. *Journal of Proteome Research*, 13(5):2262–2271.

Ma, K., Vitek, O., and Nesvizhskii, A. I. (2012). A statistical model-building perspective to identification of MS/MS spectra with PeptideProphet. *BMC Bioinformatics*, 13(S16).

MacLean, B., Eng, J. K., Beavis, R. C., and McIntosh, M. (2006). General framework for developing and evaluating database scoring algorithms using the TANDEM search engine. *Bioinformatics*, 22(22):2830–2832.

Martiánez-Bartolomé, S., Navarro, P., Martián-Maroto, F., López-Ferrer, D., Ramos-Fernández, A., Villar, M., Garciáa-Ruiz, J. P., and Vázquez, J. (2008). Properties of average score distributions of SEQUEST. *Molecular & Cellular Proteomics*, 7(6):1135–1145.

Matrix Science Ltd (2010). Mind your p's and q's: Maximising sensitivity with percolator. In *ASMS Workshop and User Meeting*.

Meyer, J. G. (2021). Deep learning neural network tools for proteomics. *Cell Reports Methods*, 1(2):100003.

Michalski, A., Cox, J., and Mann, M. (2011). More than 100, 000 detectable peptide species elute in single shotgun proteomics runs but the majority is inaccessible to data-dependent LC-MS/MS. *Journal of Proteome Research*, 10(4):1785–1793.

Neta, P., Farahani, M., Simón-Manso, Y., Liang, Y., Yang, X., and Stein, S. E. (2014). Unexpected peaks in tandem mass spectra due to reaction of product ions with residual water in mass spectrometer collision cells. *Rapid Communications in Mass Spectrometry*, 28(23):2645–2660.

Önder, O., Shao, W., Lam, H., and Brisson, D. (2014). Tracking the sources of blood meals of parasitic arthropods using shotgun proteomics and unidentified tandem mass spectral libraries. *Nature Protocols*, 9(4):842–850.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Perkins, D. N., Pappin, D. J. C., Creasy, D. M., and Cottrell, J. S. (1999). Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, 20(18):3551–3567.

Pirttilä, K., Balgoma, D., Rainer, J., Pettersson, C., Hedeland, M., and Brunius, C. (2022). Comprehensive peak characterization (cpc) in untargeted lc–ms analysis. *Metabolites*, 12(2):137.

Platt, J. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. *Advances in Kernel Methods-Support Vector Learning*, 208.

Reid, G. E. and McLuckey, S. A. (2002). Top down protein characterization via tandem mass spectrometry. *Journal of Mass Spectrometry*, 37(7):663–675.

Rios, L. M. and Sahinidis, N. V. (2012). Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293.

Rokach, L. and Maimon, O. (2005). Top-down induction of decision trees classifiers—a survey. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 35(4):476–487.

Röst, H. L., Schmitt, U., Aebersold, R., and Malmström, L. (2015). Fast and efficient xml data access for next-generation mass spectrometry. *PLOS ONE*, 10(4):e0125108.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E. and Mcclelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, pages 318–362. MIT Press, Cambridge, MA.

Salakhutdinov, R. and Hinton, G. (2009). Deep boltzmann machines. In van Dyk, D. and Welling, M., editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 448–455, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA. PMLR.

Savitski, M. M., Wilhelm, M., Hahne, H., Kuster, B., and Bantscheff, M. (2015). A scalable approach for protein false discovery rate estimation in large proteomic data sets. *Molecular & Cellular Proteomics*, 14(9):2394–2404.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Schoenholz, S. S., Hackett, S., Deming, L., Melamud, E., Jaitly, N., McAllister, F., O'Brien, J., Dahl, G., Bennett, B., Dai, A. M., and Koller, D. (2018a). Peptide-spectra matching from weak supervision.

Schoenholz, S. S., Hackett, S., Deming, L., Melamud, E., Jaitly, N., McAllister, F. E., O'Brien, J. J., Dahl, G. E., Bennett, B. D., Dai, A. M., and Koller, D. (2018b). Peptide-spectra matching with weak supervision. *arXiv: Quantitative Methods*.

Serpa, J. J., Parker, C. E., Petrotchenko, E. V., Han, J., Pan, J., and Borchers, C. H. (2012). Mass spectrometry-based structural proteomics. *European Journal of Mass Spectrometry*, 18(2):251–267.

Shah, A. R., Davidson, J., Monroe, M. E., Mayampurath, A. M., Danielson, W. F., Shi, Y., Robinson, A. C., Clowers, B. H., Belov, M. E., Anderson, G. A., and Smith, R. D. (2010). An efficient data format for mass spectrometry-based proteomics. *Journal of the American Society for Mass Spectrometry*, 21(10):1784–1788.

Shao, W., Zhu, K., and Lam, H. (2013). Refining similarity scoring to enable decoy-free validation in spectral library searching. *PROTEOMICS*, 13(22):3273–3283.

Shin, J., Lee, W., and Lee, W. (2008). Structural proteomics by NMR spectroscopy. *Expert Review of Proteomics*, 5(4):589–601.

Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1).

Shteynberg, D., Nesvizhskii, A. I., Moritz, R. L., and Deutsch, E. W. (2013). Combining results of multiple search engines in proteomics. *Molecular & Cellular Proteomics*, 12(9):2383–2393.

Siegelmann, H. T. and Sontag, E. D. (1992). On the computational power of neural nets. In *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*. ACM Press.

Spivak, M., Weston, J., Bottou, L., Käll, L., and Noble, W. S. (2009). Improvements to the percolator algorithm for peptide identification from shotgun proteomics data sets. *Journal of Proteome Research*, 8(7):3737–3745.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Highway networks. *CoRR*, abs/1505.00387.

Stanford, T. E., Bagley, C. J., and Solomon, P. J. (2016). Informed baseline subtraction of proteomic mass spectrometry data aided by a novel sliding window algorithm. *Proteome Science*, 14(1).

Stein, S. (2008). Nist libraries of peptide fragmentation mass spectra, nist standard reference database 1 c.

Tabb, D. L. (2015). The SEQUEST family tree. *Journal of the American Society for Mass Spectrometry*, 26(11):1814–1819.

Tariq, M. U. and Saeed, F. (2021). Specollate: Deep cross-modal similarity network for mass spectrometry data based peptide deductions. *PLOS ONE*, 16(10):e0259349.

Teo, G. C., Polasky, D. A., Yu, F., and Nesvizhskii, A. I. (2020). Fast deisotoping algorithm and its implementation in the MSFragger search engine. *Journal of Proteome Research*, 20(1):498–505.

The, M., MacCoss, M. J., Noble, W. S., and Käll, L. (2016). Fast and accurate protein false discovery rates on large-scale proteomics data sets with percolator 3.0. *Journal of the American Society for Mass Spectrometry*, 27(11):1719–1727.

Tiwary, S., Levy, R., Gutenbrunner, P., Soto, F. S., Palaniappan, K. K., Deming, L., Berndl, M., Brant, A., Cimermancic, P., and Cox, J. (2019). High-quality MS/MS spectrum prediction for data-dependent and data-independent acquisition data analysis. *Nature Methods*, 16(6):519–525.

Tran, N. H., Zhang, X., Xin, L., Shan, B., and Li, M. (2017). De novo peptide sequencing by deep learning. *Proceedings of the National Academy of Sciences*, 114(31):8247–8252.

Tu, C., Sheng, Q., Li, J., Ma, D., Shen, X., Wang, X., Shyr, Y., Yi, Z., and Qu, J. (2015). Optimization of search engines and postprocessing approaches to maximize peptide and protein identification for high-resolution mass data. *Journal of Proteome Research*, 14(11):4662–4673.

Urban, P. L. (2016). Quantitative mass spectrometry: an overview. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2079):20150382.

Välikangas, T., Suomi, T., and Elo, L. L. (2017). A comprehensive evaluation of popular proteomics software workflows for label-free proteome quantification and imputation. *Briefings in Bioinformatics*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.

Vu, T. N., Mrzic, A., Valkenborg, D., Maes, E., Lemière, F., Goethals, B., and Laukens, K. (2014). Unravelling associations between unassigned mass spectrometry peaks with frequent itemset mining techniques. *Proteome Science*, 12(1).

Wang, G., Wu, W. W., Zhang, Z., Masilamani, S., and Shen, R.-F. (2008). Decoy methods for assessing false positives and false discovery rates in shotgun proteomics. *Analytical Chemistry*, 81(1):146–159.

Wang, J., Pérez-Santiago, J., Katz, J. E., Mallick, P., and Bandeira, N. (2010). Peptide identification from mixture tandem mass spectra. *Molecular & Cellular Proteomics*, 9(7):1476–1485.

Wang, J., Zhang, Y., and Yu, Y. (2015). Crescendo: A protein sequence database search engine for tandem mass spectra. *Journal of the American Society for Mass Spectrometry*, 26(7):1077–1084.

Wang, M., Wang, J., Carver, J., Pullman, B. S., Cha, S. W., and Bandeira, N. (2018). Assembling the community-scale discoverable human proteome. *Cell Systems*, 7(4):412–421.e5.

Wen, B., Zeng, W.-F., Liao, Y., Shi, Z., Savage, S. R., Jiang, W., and Zhang, B. (2020). Deep learning in proteomics. *PROTEOMICS*, 20(21-22):1900335.

Wilhelm, M., Zolg, D. P., Graber, M., Gessulat, S., Schmidt, T., Schnatbaum, K., Schwencke-Westphal, C., Seifert, P., de Andrade Krätzig, N., Zerweck, J., Knaute, T., Bräunlein, E., Samaras, P., Lautenbacher, L., Klaeger, S., Wenschuh, H., Rad, R., Delanghe, B., Huhmer, A., Carr, S. A., Clauser, K. R., Krackhardt, A. M., Reimer, U., and Kuster, B. (2021). Deep learning boosts sensitivity of mass spectrometry-based immunopeptidomics. *Nature Communications*, 12(1).

Wu, G., Wan, X., and Xu, B. (2018). A new estimation of protein-level false discovery rate. *BMC Genomics*, 19(S6).

Wysocki, V. H., Resing, K. A., Zhang, Q., and Cheng, G. (2005). Mass spectrometry of peptides and proteins. *Methods*, 35(3):211–222.

Xu, M., Li, Z., and Li, L. (2013). Combining percolator with x!tandem for accurate and sensitive peptide identification. *Journal of Proteome Research*, 12(6):3026–3033.

Xu, Y. and Goodacre, R. (2018). On splitting training and validation set: A comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. *Journal of Analysis and Testing*, 2(3):249–262.

Yen, C.-Y., Houel, S., Ahn, N. G., and Old, W. M. (2011). Spectrum-to-spectrum searching using a proteome-wide spectral library. *Molecular & Cellular Proteomics*, 10(7):M111.007666.

Yuan, Z.-F., Lin, S., Molden, R. C., and Garcia, B. A. (2014). Evaluation of proteomic search engines for the analysis of histone modifications. *Journal of Proteome Research*, 13(10):4470–4478.

Zeng, W.-F., Zhou, X.-X., Willems, S., Ammar, C., Wahle, M., Bludau, I., Voytik, E., Strauss, M. T., and Mann, M. (2022). AlphaPeptDeep: a modular deep learning framework to predict peptide properties for proteomics. *Nature Communications*, 13(1).

Zhang, J., Li, J., Liu, X., Xie, H., Zhu, Y., and He, F. (2008). A nonparametric model for quality control of database search results in shotgun proteomics. *BMC Bioinformatics*, 9(1).

Zhang, J., Xin, L., Shan, B., Chen, W., Xie, M., Yuen, D., Zhang, W., Zhang, Z., Lajoie, G. A., and Ma, B. (2011). PEAKS DB:de NovoSequencing assisted database search for sensitive and accurate peptide identification. *Molecular & Cellular Proteomics*, 11(4):M111.010587.

Zhang, J., Xin, L., Shan, B., Chen, W., Xie, M., Yuen, D., Zhang, W., Zhang, Z., Lajoie, G. A., and Ma, B. (2012). PEAKS DB: De novo sequencing assisted database search for sensitive and accurate peptide identification. *Molecular & Cellular Proteomics*, 11(4):M111.010587.

Zhang, Y., Fonslow, B. R., Shan, B., Baek, M.-C., and Yates, J. R. (2013). Protein analysis by shotgun/bottom-up proteomics. *Chemical Reviews*, 113(4):2343–2394.

Zhang, Y., Xu, T., Shan, B., Hart, J., Aslanian, A., Han, X., Zong, N., Li, H., Choi, H., Wang, D., Acharya, L., Du, L., Vogt, P. K., Ping, P., and Yates, J. R. (2015). Proteininferencer: Confident protein identification and multiple experiment comparison for large scale proteomics projects. *Journal of Proteomics*, 129:25–32.

Zolg, D. P., Wilhelm, M., Schnatbaum, K., Zerweck, J., Knaute, T., Delanghe, B., Bailey, D. J., Gessulat, S., Ehrlich, H.-C., Weininger, M., Yu, P., Schlegl, J., Kramer, K., Schmidt, T., Kusebauch, U., Deutsch, E. W., Aebersold, R., Moritz, R. L., Wenschuh, H., Moehring, T., Aiche, S., Huhmer, A., Reimer, U., and Kuster, B. (2017). Building ProteomeTools based on a complete synthetic human proteome. *Nature Methods*, 14(3):259–262.

Appendix A

Pepid

A.1. Code Organization

A.1.1. `main.py`

Entry point. A search may be launched with `python pepid.py config.cfg`, `import peppid; pepid.run("config.cfg")` or `python -mpepid config.cfg` where `config.cfg` is the user configuration file. This will first read a default configuration file at `data/default.cfg` at the local path to fill in all necessary parameters with sensible defaults, then read the user configuration file to overwrite any options the user may like to change. The program will then read the completed configuration data and launch the required task for the various enabled pipeline steps, such as launching the `pepid_search` module's `run` function to perform the preprocessing, search, and results extraction (again based on user configuration for these steps).

A.1.2. `pepid_search.py`

Operates the "search" part of the pipeline, including preprocessing, user postprocessing, and PSM scoring (see below for more details on each step).

A.1.3. `search.py`

The `search` module handles the search proper. It invokes the scoring function on candidate peptide and spectrum pairs. Multiprocessing dispatches on batches of candidates within a user-specified tolerance of the precursor neutral mass of the given query, one query at a time. All matches are saved in the final results database so long as the score has a value higher than 0. It is then possible to extract the top N scores (with a user-specified N) to a tsv file for further analysis, or to use the database itself in any way required for the current experiment, including performing rescoring on the entire set of results instead of just the top N results.

A.1.4. queries.py

Query processing functions are contained in this file. Multiprocessing dispatches on query batches.

A.1.5. db.py

Functions relevant to processing the input fasta database are contained in `db.py`. This includes database processing and input into the temporary database as well as helper functions to count peptides and in-silico digestion and post-translational modification (PTM) generation. Multiprocessing dispatches on batches of protein sequences for in-silico digestion and PTM generation, and on batches of peptides for further processing.

A.1.6. blackboard.py

This module contains the behavior that must repeatedly be invoked upon process creation (see: `*node.py` below) as well as shared context state such as the configuration data for the program. It contains functions to prepare databases and database connections.

A.1.7. pepid_mp.py

Due to various bugs and misfeatures in available multiprocessing packages for python, a custom solution was developed based on unix `select` and `socket`. This solution also presents several advantages such as better control over input and output, explicitly clean working process memory (each process is started from scratch without forking or copying working memory over to the new process), explicit protocols to relay information to and from the slave nodes back to the master node, and better task status identification for task monitory due to the explicit script name and parameters during task launch.

A.1.8. {search_,queries_,db_,pin_,output_}node.py

Files implementing the actual multiprocessing behavior specific to each task. As the names imply, `node.py` is the base behavior for the system that contains boilerplate for launch (e.g. it decodes basic launch message codes and calls the specific node behavior for startup and shutdown) and rescue (currently, it shuttles error messages back to the master node if any error happens during subprocess execution) and relegates processing to the specific nodes. Other `*_node.py` files contain the specific behavior for proper search, queries processing, and input database processing, and output (to percolator input, PIN, format, or to tsv output format), respectively.

A.1.9. pepid_io.py

Contains output processing functions, mostly to extract the top N results (with N as specified in user configuration) and output them to a tsv file for downstream processing such as direct application of percolator or other rescoring tools.

A.1.10. `pepid_rescore.py`

Implements the basic rescoring workflow, dispatching the rescoring and output tasks to the appropriate user module (in the default release, that is the `pepid_percolator.py` or `pepid_randomforest.py` modules, as described in the user configuration file).

A.1.11. `pepid_percolator.py`

A wrapper around percolator for rescoring. Also handles Pepid TSV to percolator IN (PIN) file conversion.

A.1.12. `pepid_rf_finetune.py`

Implements a custom rescoring algorithm based on random forests. Relies on a pretrained segment which can be regenerated using the `train_rf_rescorer.py` utility script in the `ml` directory.

A.1.13. `pepid_utils.py`

Generic utilities, such as for *in-silico* peptide fragmentation or calculating peptide mass.

A.1.14. `extensions.py`

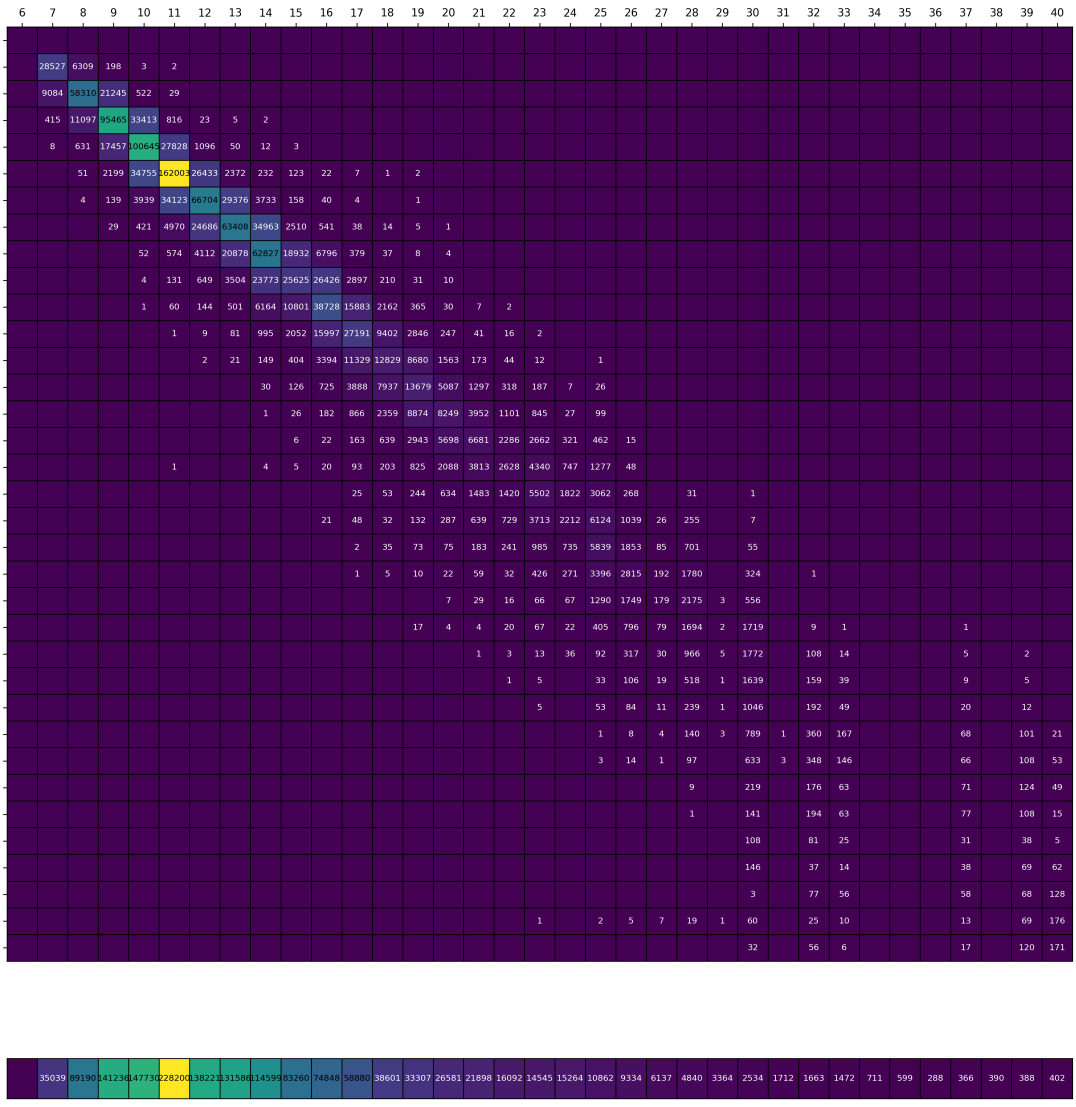
Extension facilities, including spectrum generation, length prediction, and related.

Appendix B

Length Prediction

Supplementary Material

Length Confusion Matrix



Supplementary Figure 1 – Full length prediction confusion matrix across the lengths the model was trained for (6-40 inclusive). The bottom row shows marginal counts for each class.

Model Details

The model architecture is presented in the main text. Here we describe the exact hyperparameters in the final model.

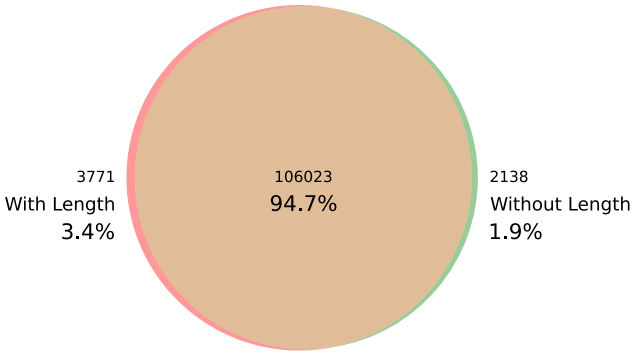
The convolutional layers in the preprocessing stack were procedurally determined using the starting dimensions of the input. In total, 13 layers, composed of a convolution, a batch normalization (batchnorm), and rectified linear (ReLU), and a pooling layer. All pooling layers were mean-pools of size and stride 2. For the convolutional layers, they had a dimension of 11, unless the input size was not compatible, in which case they were 12. The last layer had a kernel size of 1. All layers used an embedding size of 500 except the final layer, which converts the 500 latent units into our output size of 35. The pattern of kernel sizes was thus: 11, 12, 11, 12, 12, 11, 12, 11, 12, 12, 11, 11, 1. The final layer is followed by a non-linearity of log-softmax rather than a ReLU.

To convert the convolution outputs to a latent representation compatible with the prediction "head", a convolution of kernel size 1 and a pooling layer of size and stride 2 was applied.

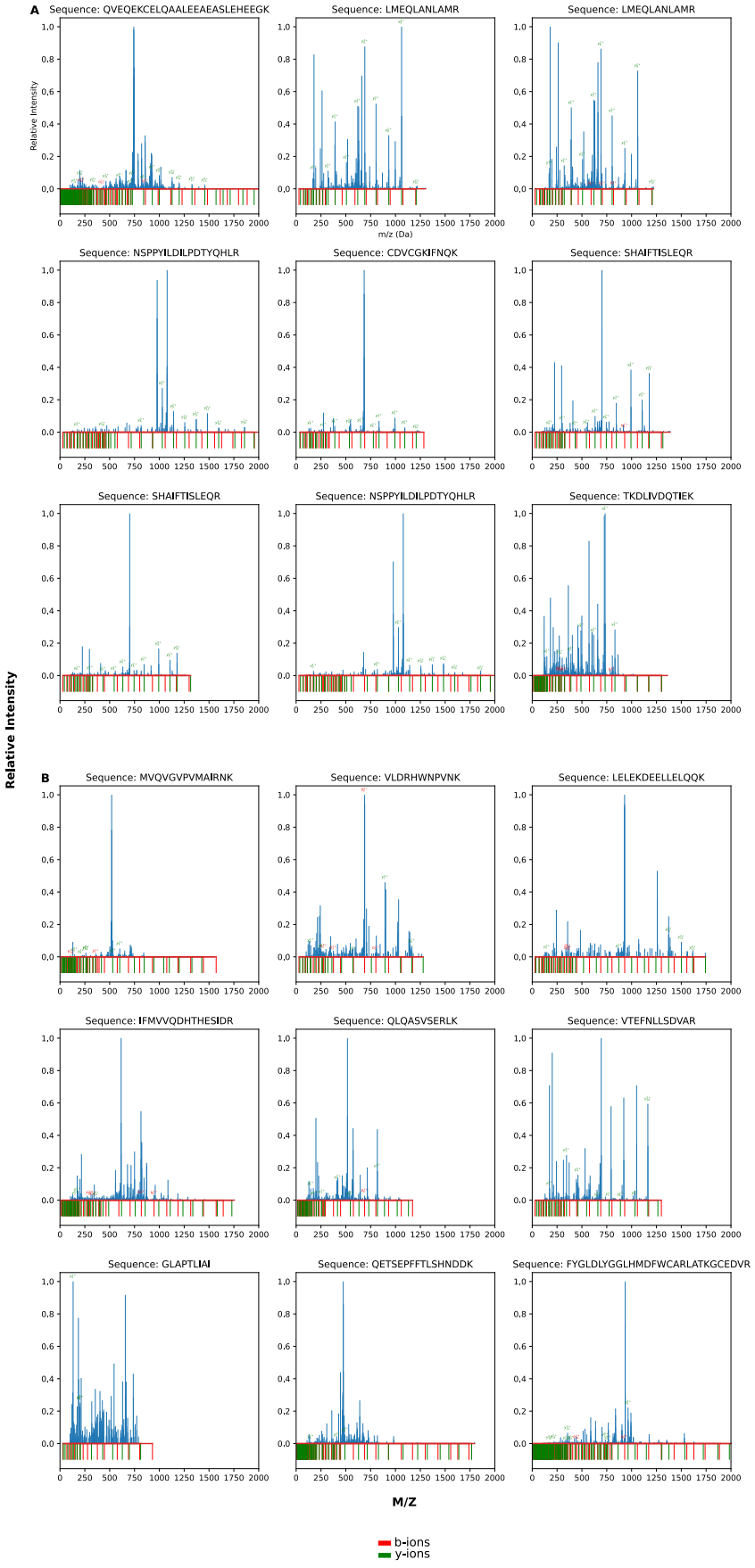
The linear layers processing the spectrum are composed of 8 fully-connected layers with 500 units in their latent spaces. The input of the first network is the whole spectrum, of size 50000. A 9th fully-connected layer converts the latent output to our output size of 35.

The linear metadata-processing network is composed of 4 fully-connected layers with 100 units each, followed by one last layer with 35 outputs. The combination network has the same topology, but while the metadata network takes just the input mass as a feature, the combination network takes 1100 features, namely the latent representations of the other networks as a concatenated vector.

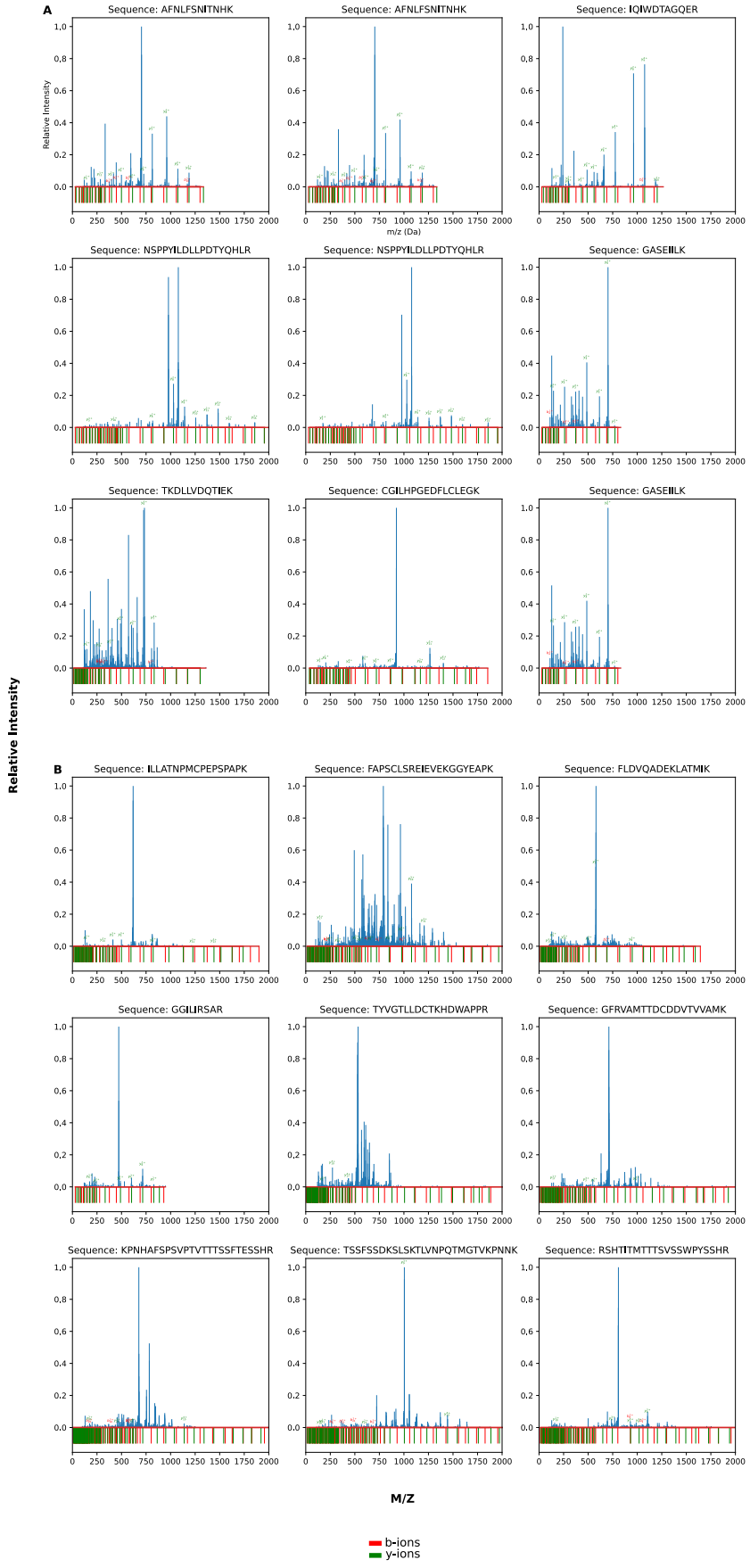
Additional Results



Supplementary Figure 2 – Venn diagram of unique peptides with and without length identified in ProteomeTools at 1% FDR



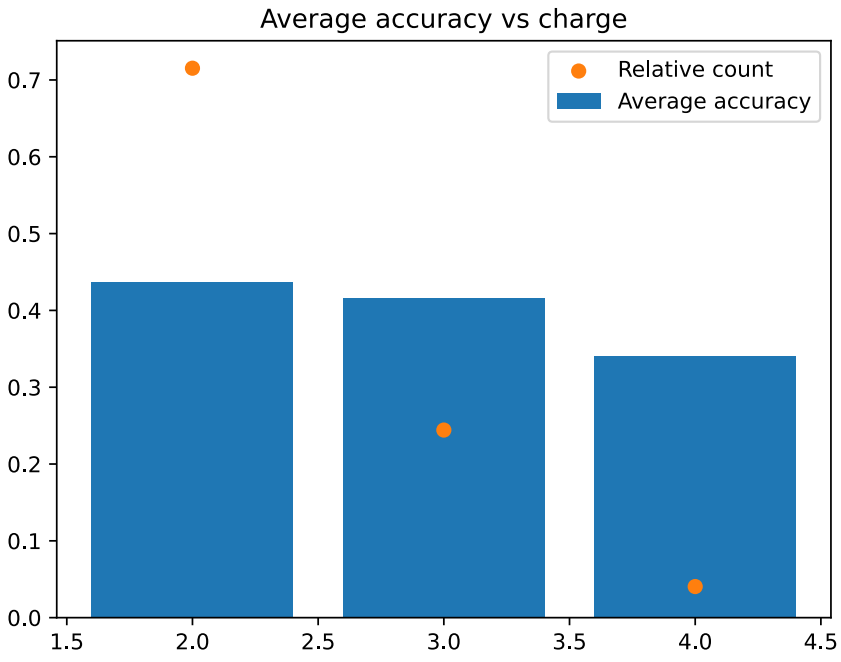
Supplementary Figure 3 – Top spectra in the set of identified PSMs past 1% FDR in ProteomeTools, without using peptide length features. **A**: high scorers not present in the length-augmented results. **B**: low scorers not present in the length-augmented results.



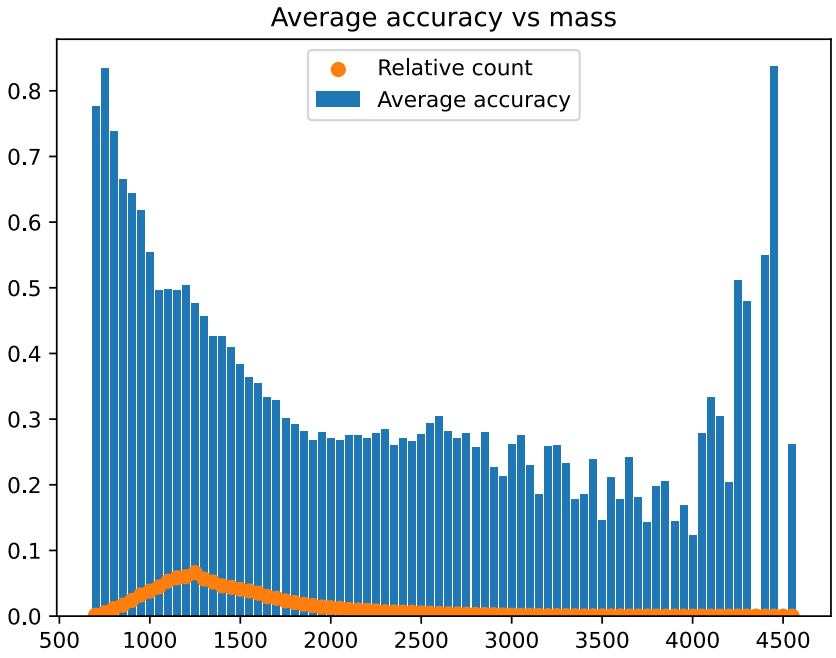
Supplementary Figure 4 – Top spectra in the set of identified PSMs past 1% FDR in ProteomeTools, using peptide length features. **A**: high scorers not present in the length-free results. **B**: low scorers not present in the length-free results.

B.1. Analysis of Length Prediction Biases

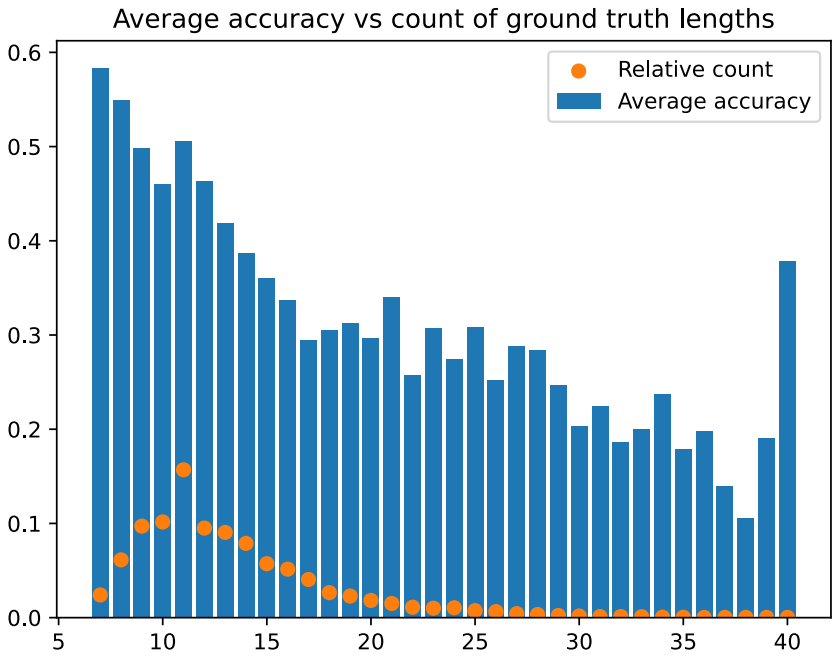
This section shows the distribution of accuracies and absolute length differences (between predicted and ground truth lengths) against several key features of peptides to assess biases. The results suggest that the main source of bias is due to data availability, but the model also performs better on shorter or lighter peptides. In the figures shown below, the orange dots are relative counts in each category (computed as the count for the category divided by total count), while the bars represent the average value in the category for the metric (either accuracy or absolute length difference).



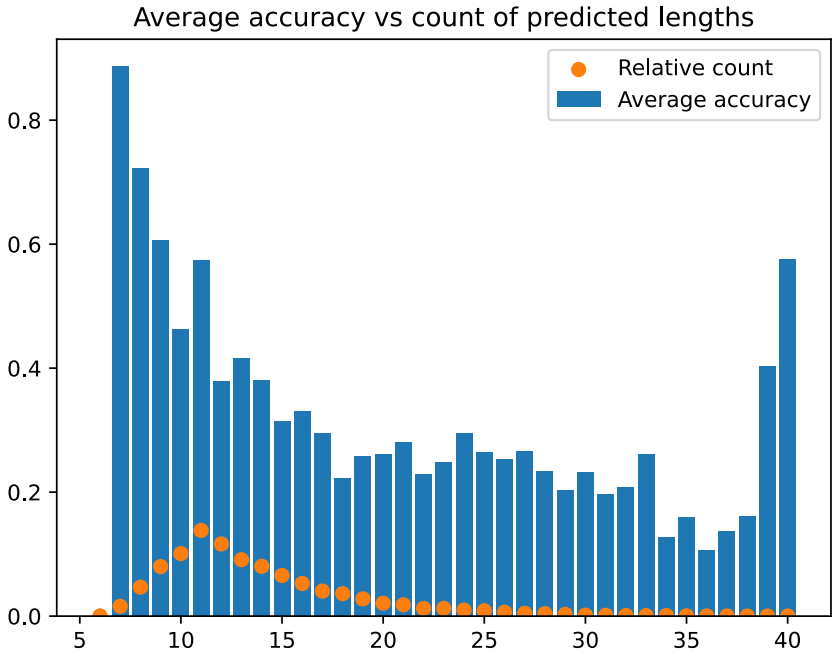
Supplementary Figure 5 – Average accuracy vs percursor charge.



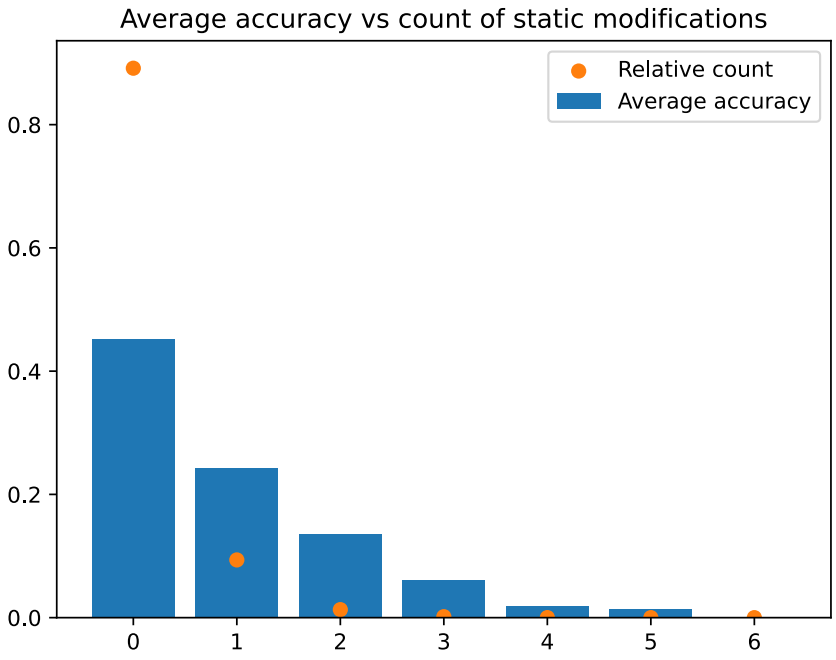
Supplementary Figure 6 – Average accuracy vs percursor mass.



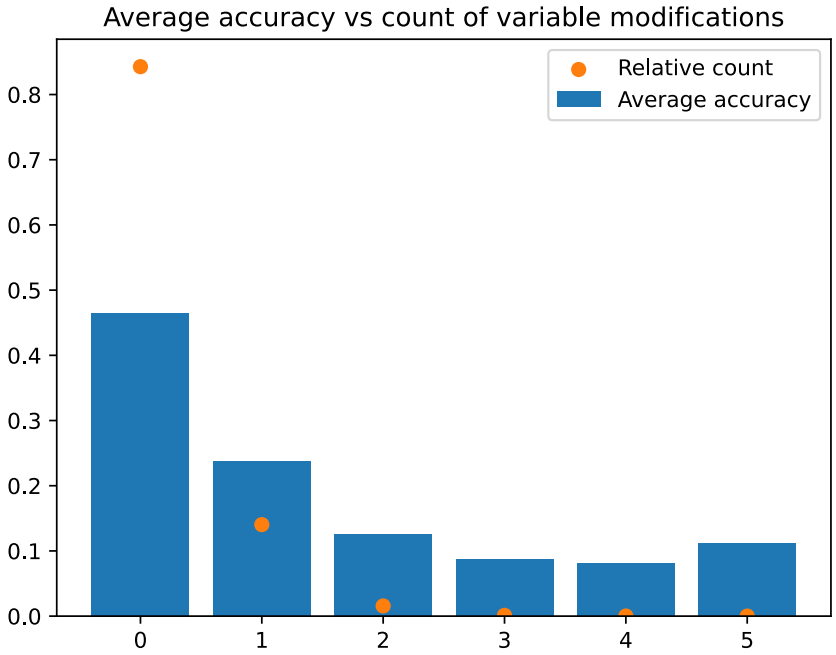
Supplementary Figure 7 – Average accuracy vs ground truth peptide length.



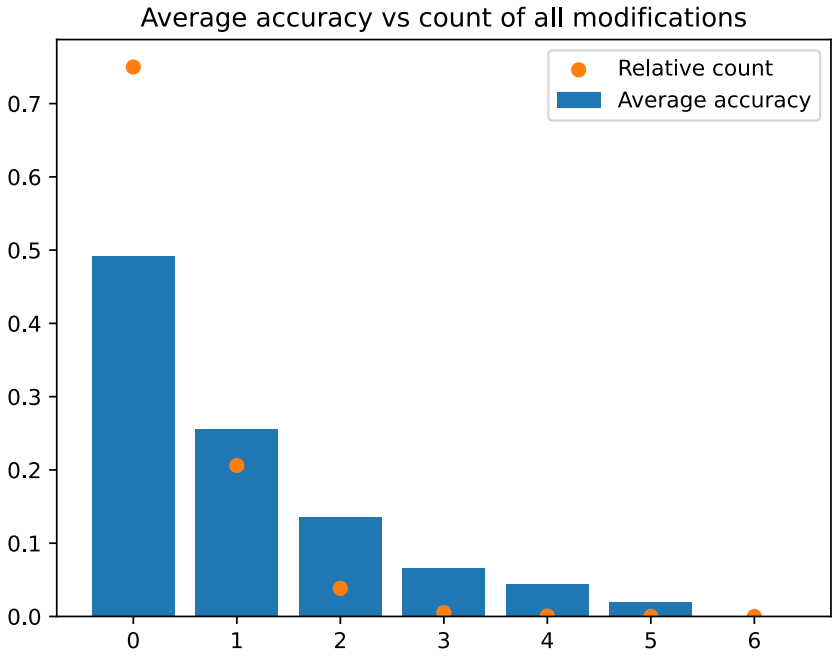
Supplementary Figure 8 – Average accuracy vs predicted peptide length.



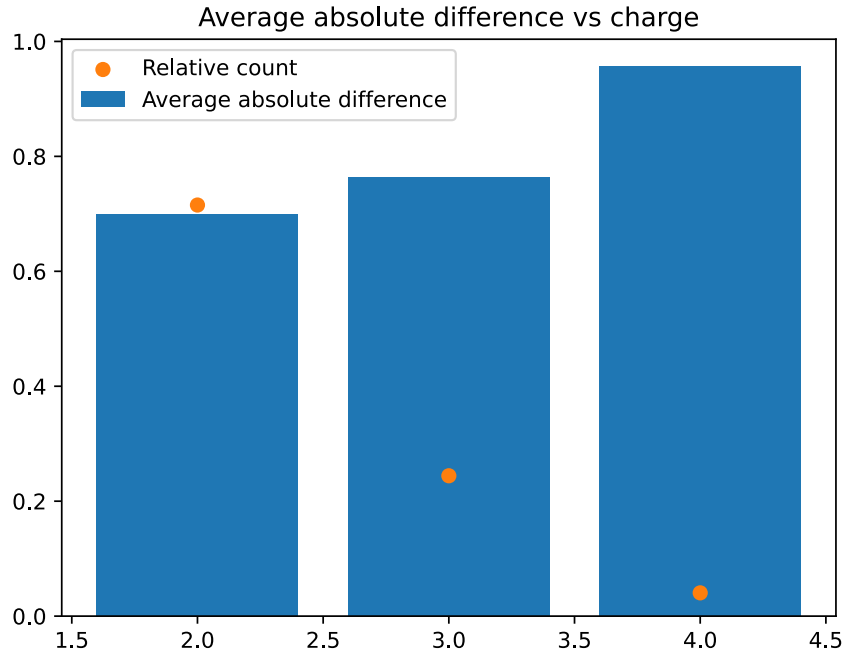
Supplementary Figure 9 – Average accuracy vs count of static modifications.



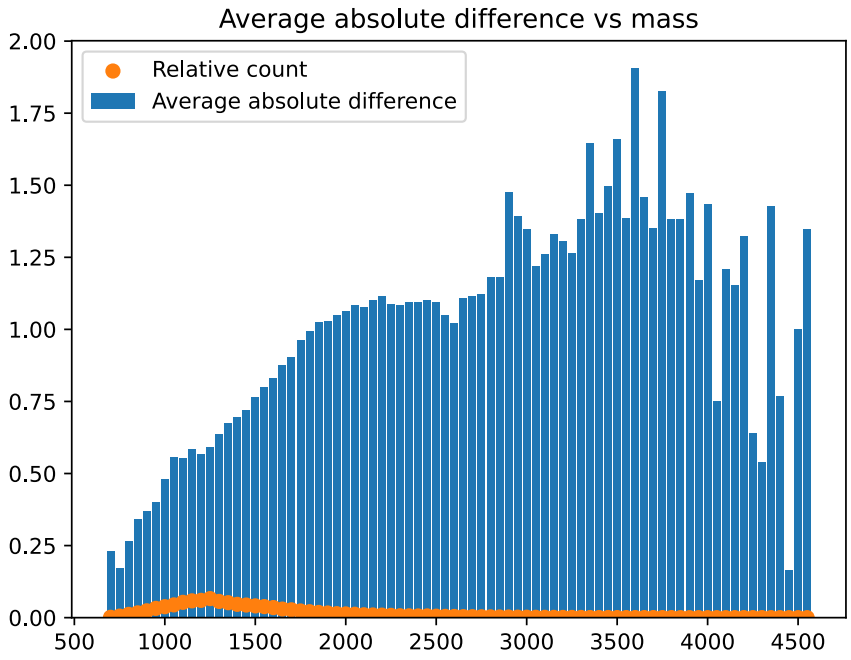
Supplementary Figure 10 – Average accuracy vs count of variable modifications.



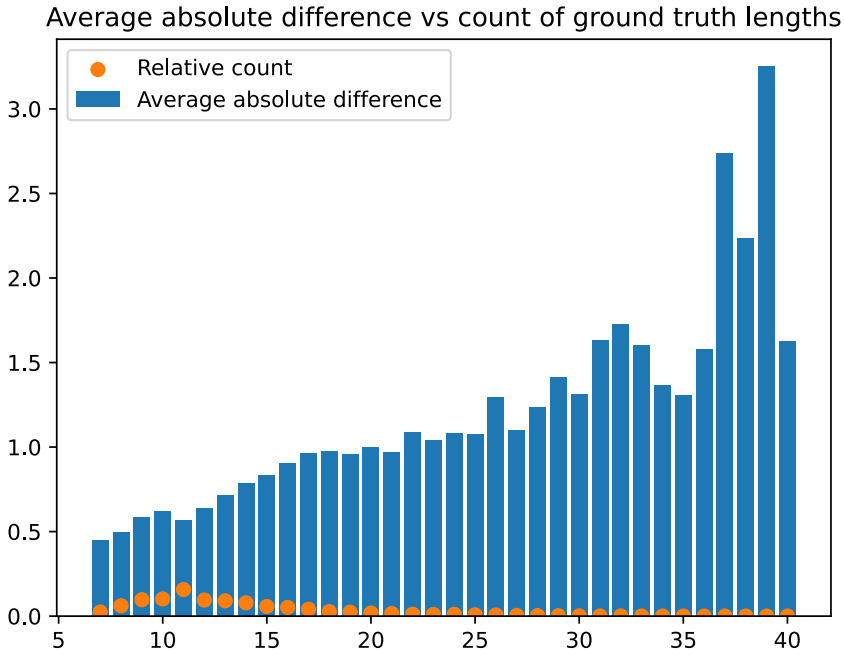
Supplementary Figure 11 – Average accuracy vs count of all (variable and static) modifications.



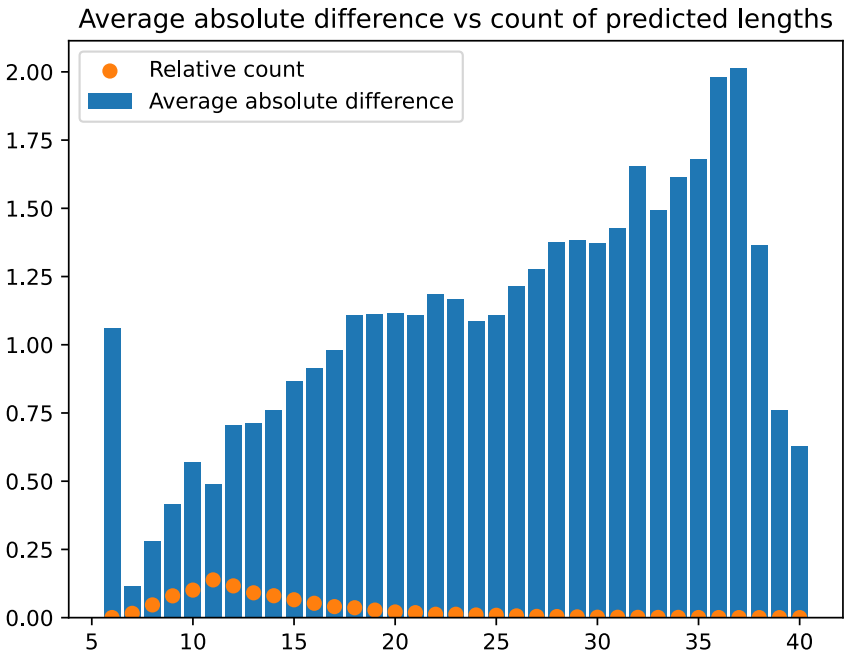
Supplementary Figure 12 – Average length difference vs percursor charge.



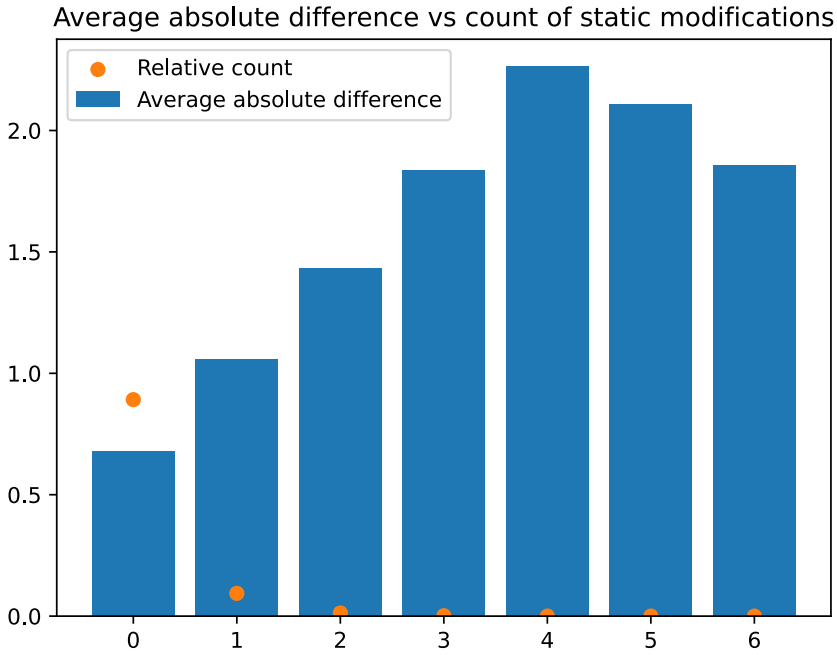
Supplementary Figure 13 – Average length difference vs percursor mass.



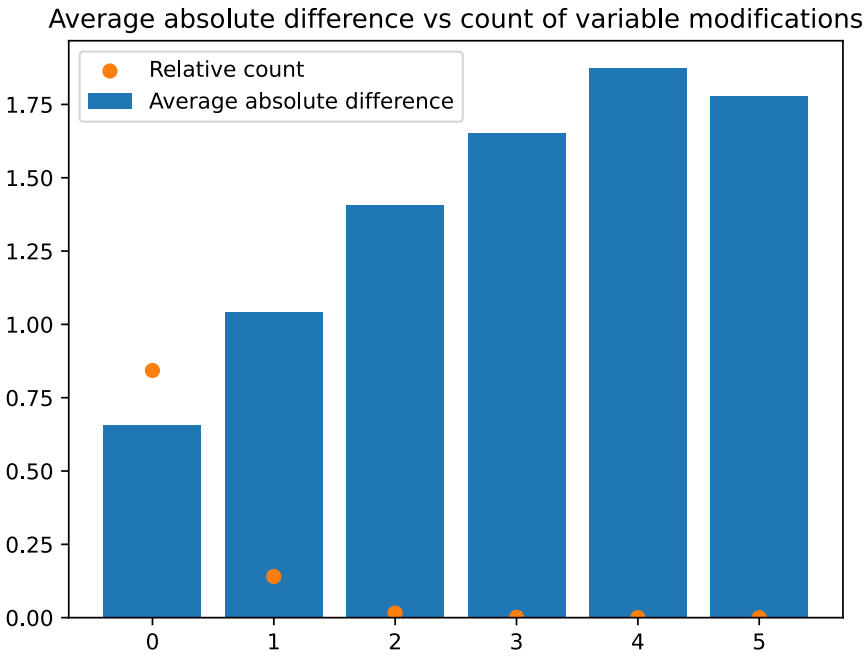
Supplementary Figure 14 – Average length difference vs ground truth peptide length.



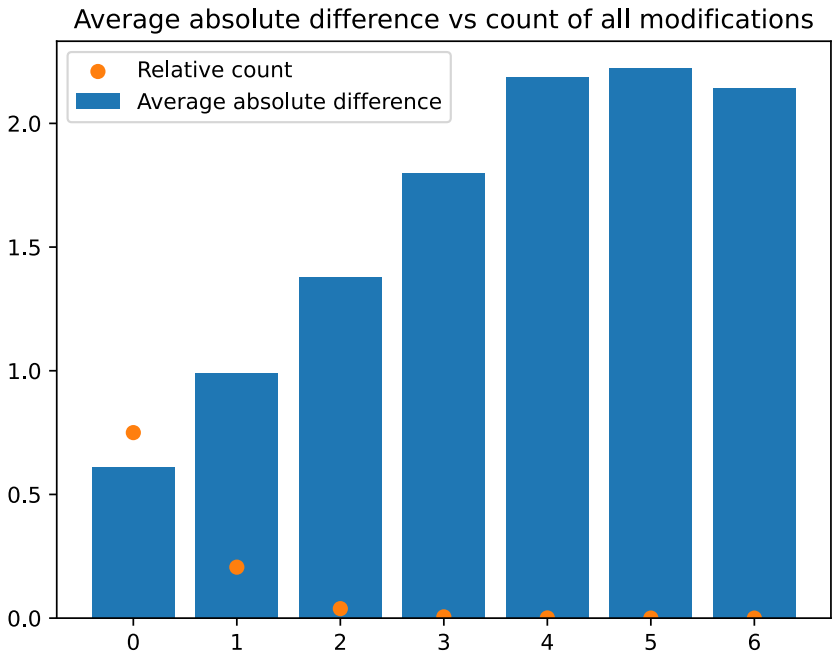
Supplementary Figure 15 – Average length difference vs predicted peptide length.



Supplementary Figure 16 – Average length difference vs count of static modifications.



Supplementary Figure 17 – Average length difference vs count of variable modifications.



Supplementary Figure 18 – Average length difference vs count of all (variable and static) modifications.

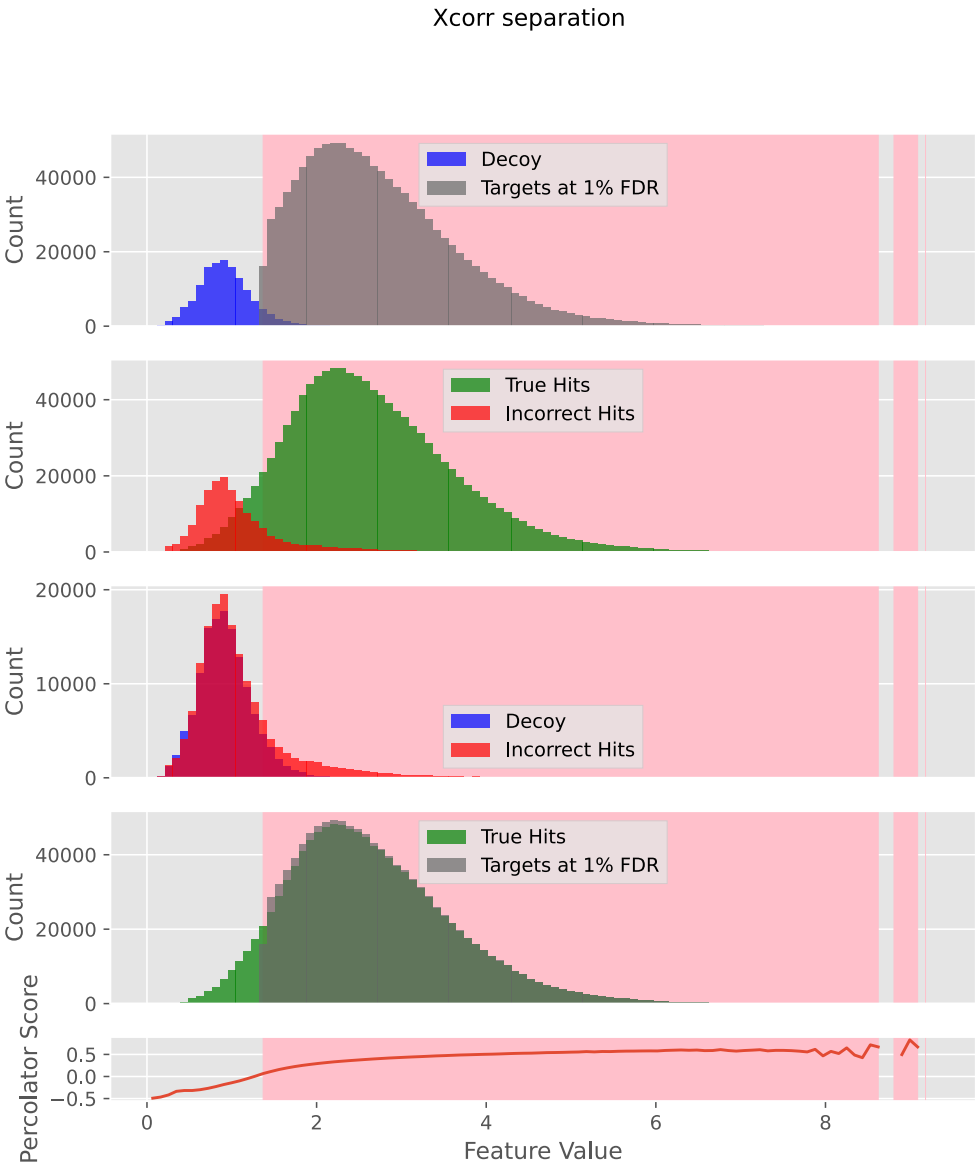
Appendix C

False Discovery Rate Estimation

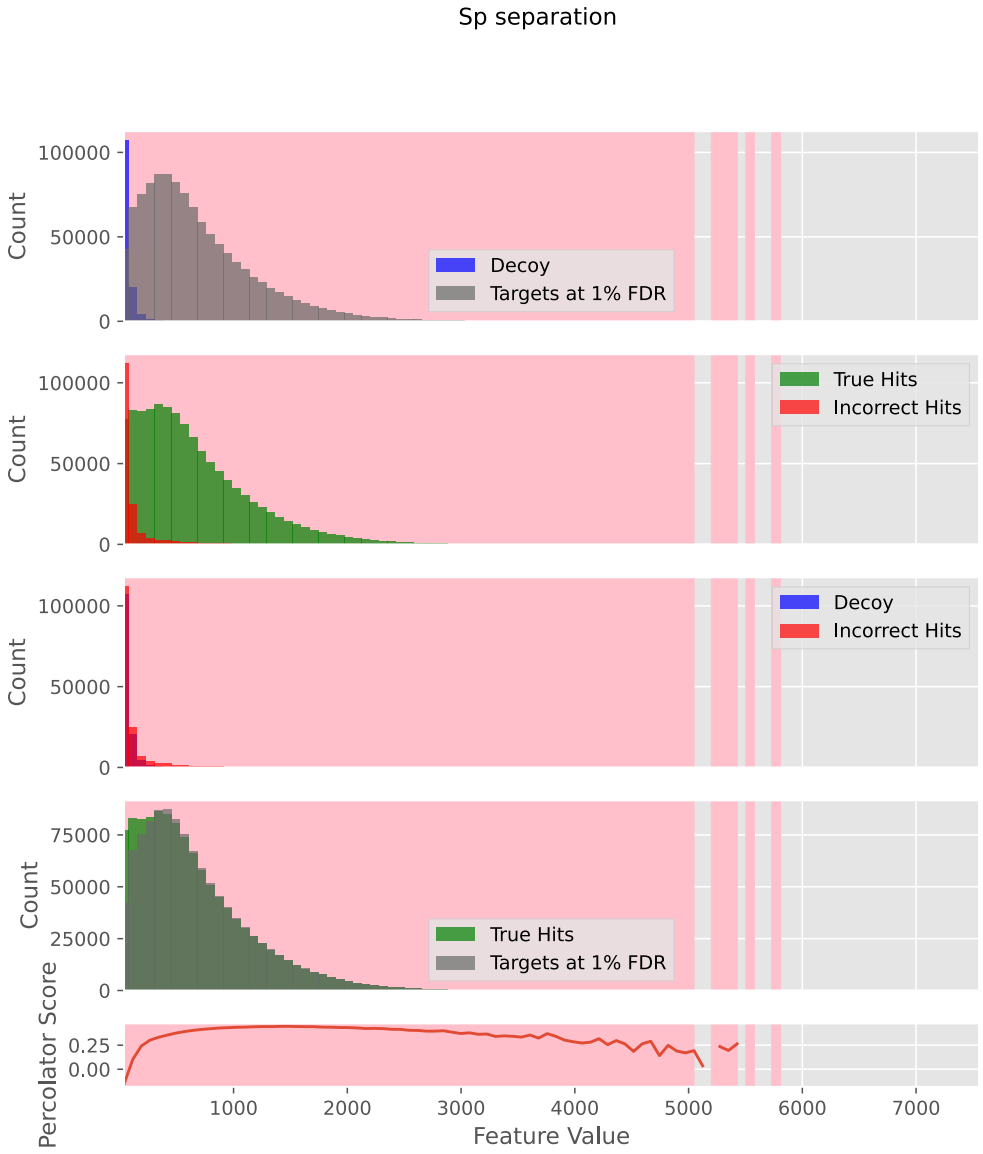
C.1. Distributional skew visualization for single features with Percolator rescoring

General figure description in this section The distribution pairs match those used by Percolator during model fitting: roughly speaking, Percolator is fit to differentiate between decoys and targets that pass a 1% FDR threshold (top), but the real metric of interest is the separation between true hits and incorrect hits (second from top). There is a slight skew between decoys and incorrect hits (third from top) as well as targets passing 1% FDR threshold and true hits (bottom). The pink zone represents values putatively categorized as true hits by Percolator score at 1% FDR. Figures represent select distributions representative of the kind of skews present in the data.

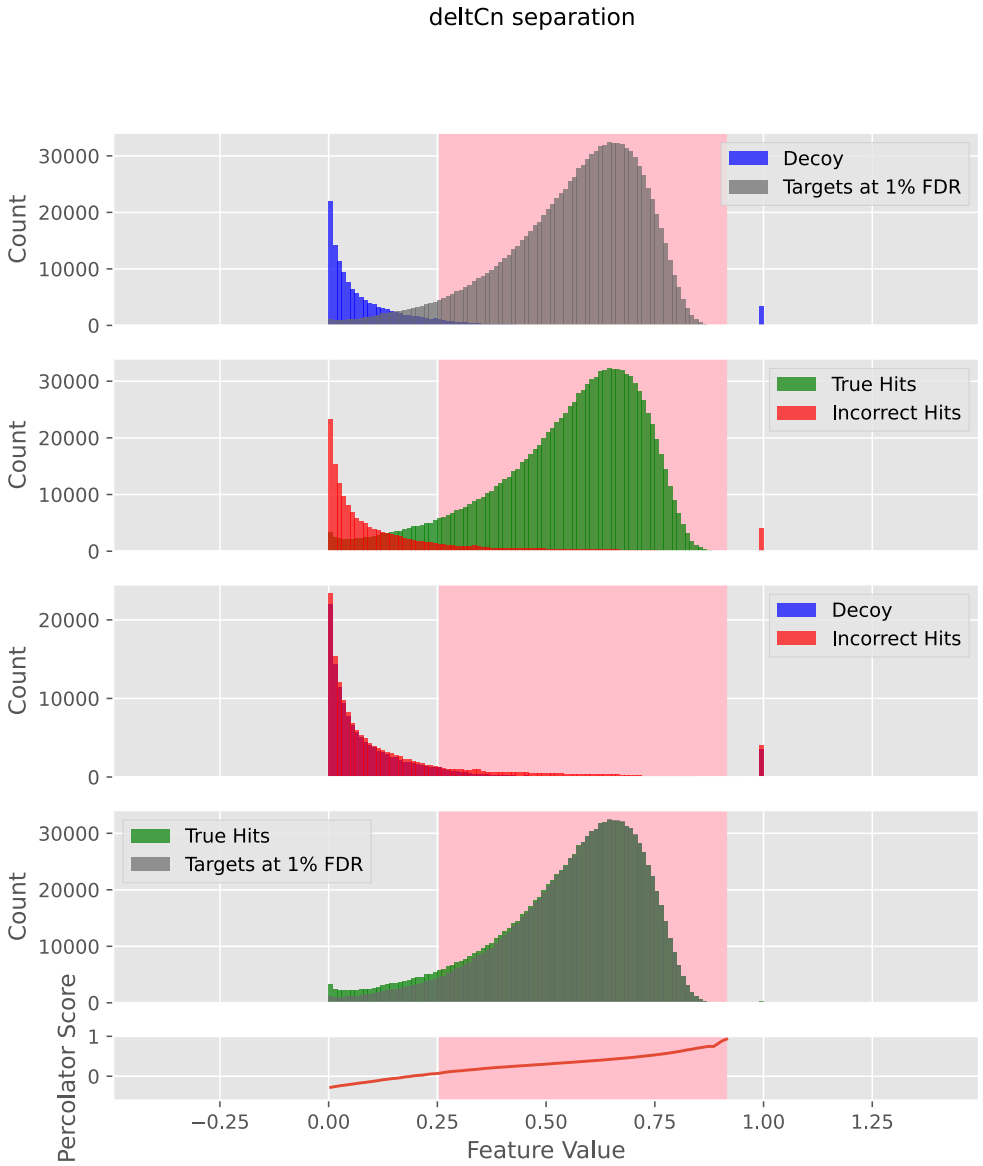
C.1.1. Distribution Skews With Comet



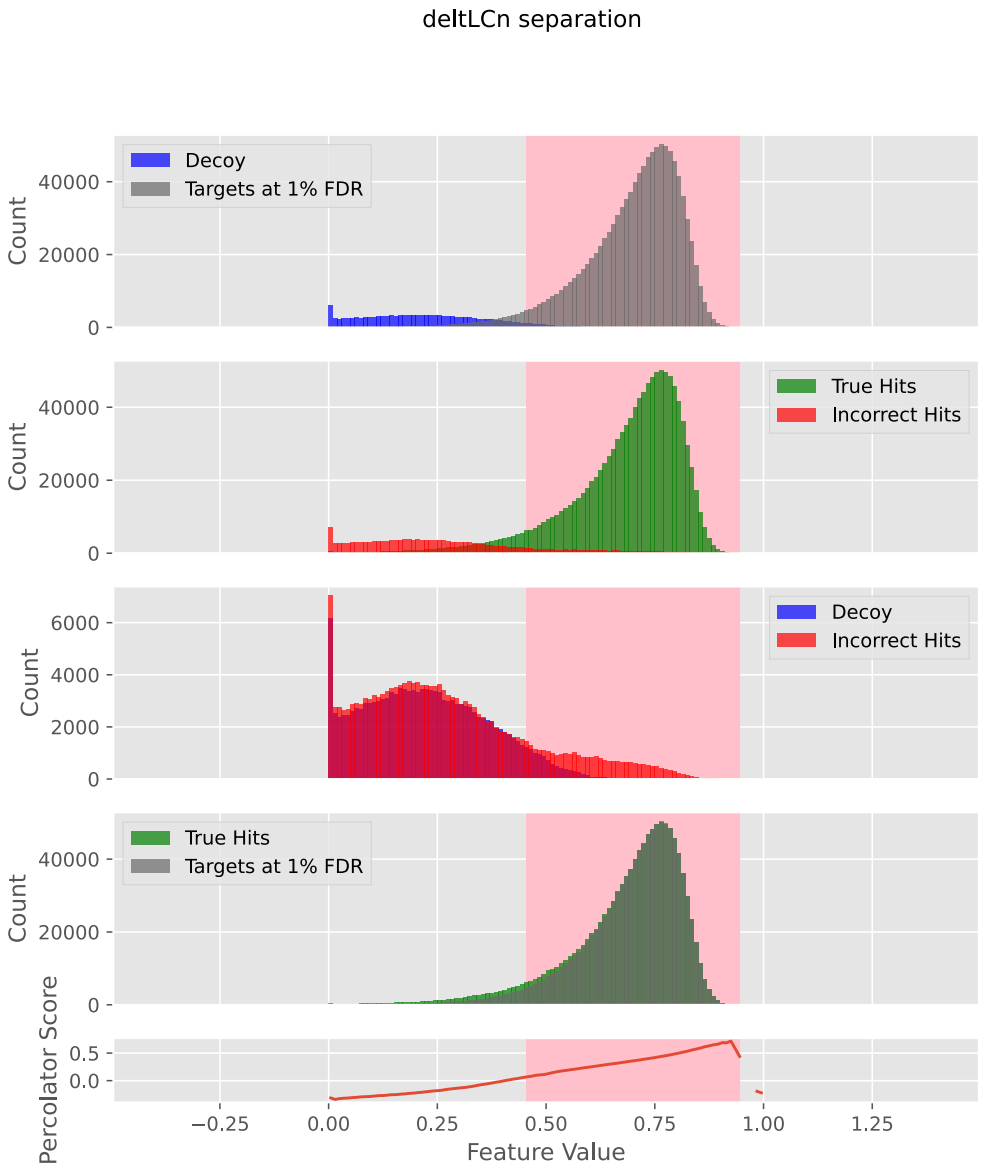
Supplementary Figure 1 – Distribution skew in ProteomeTools search results using Comet along the Xcorr feature.



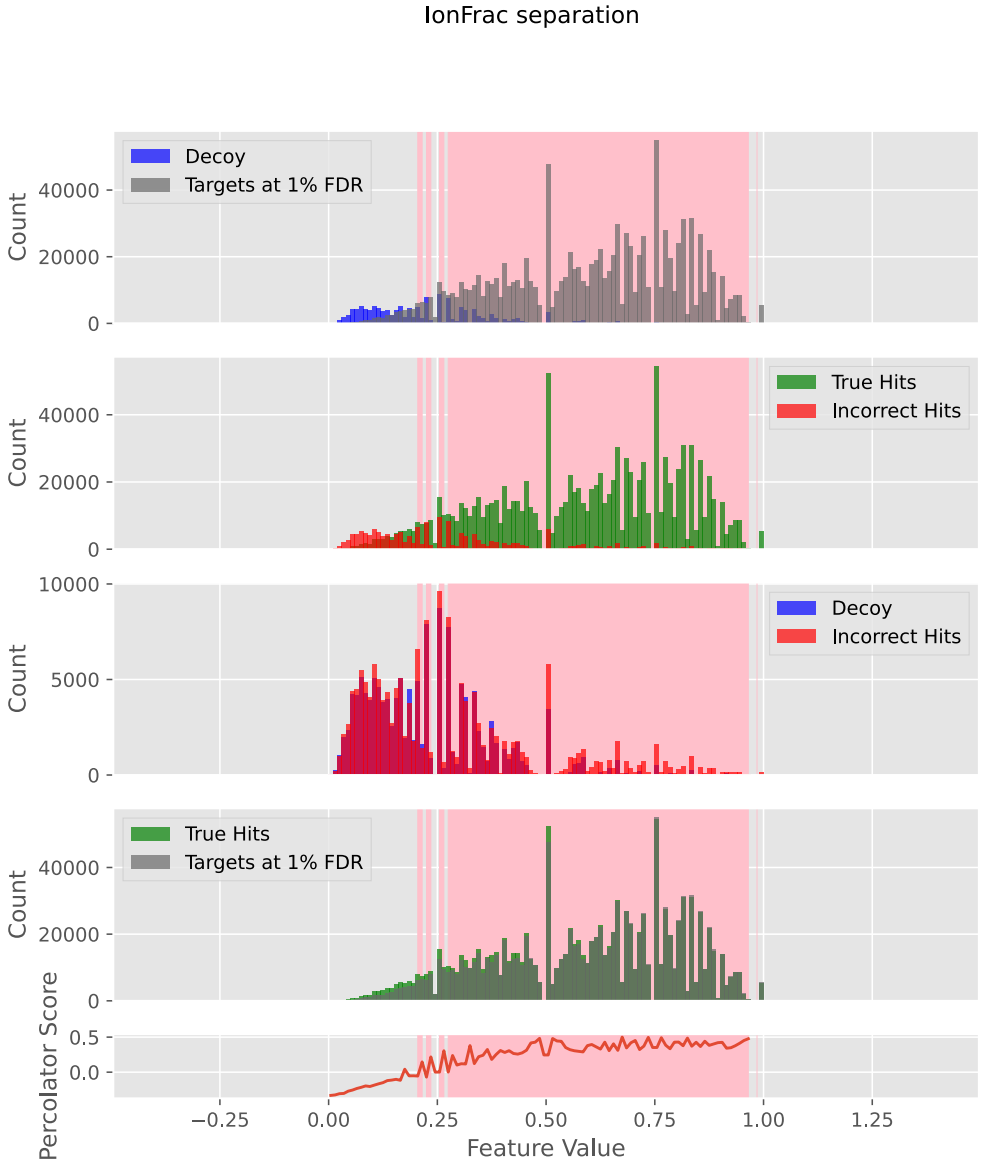
Supplementary Figure 2 – Distribution skew in ProteomeTools search results using Comet Sp feature.



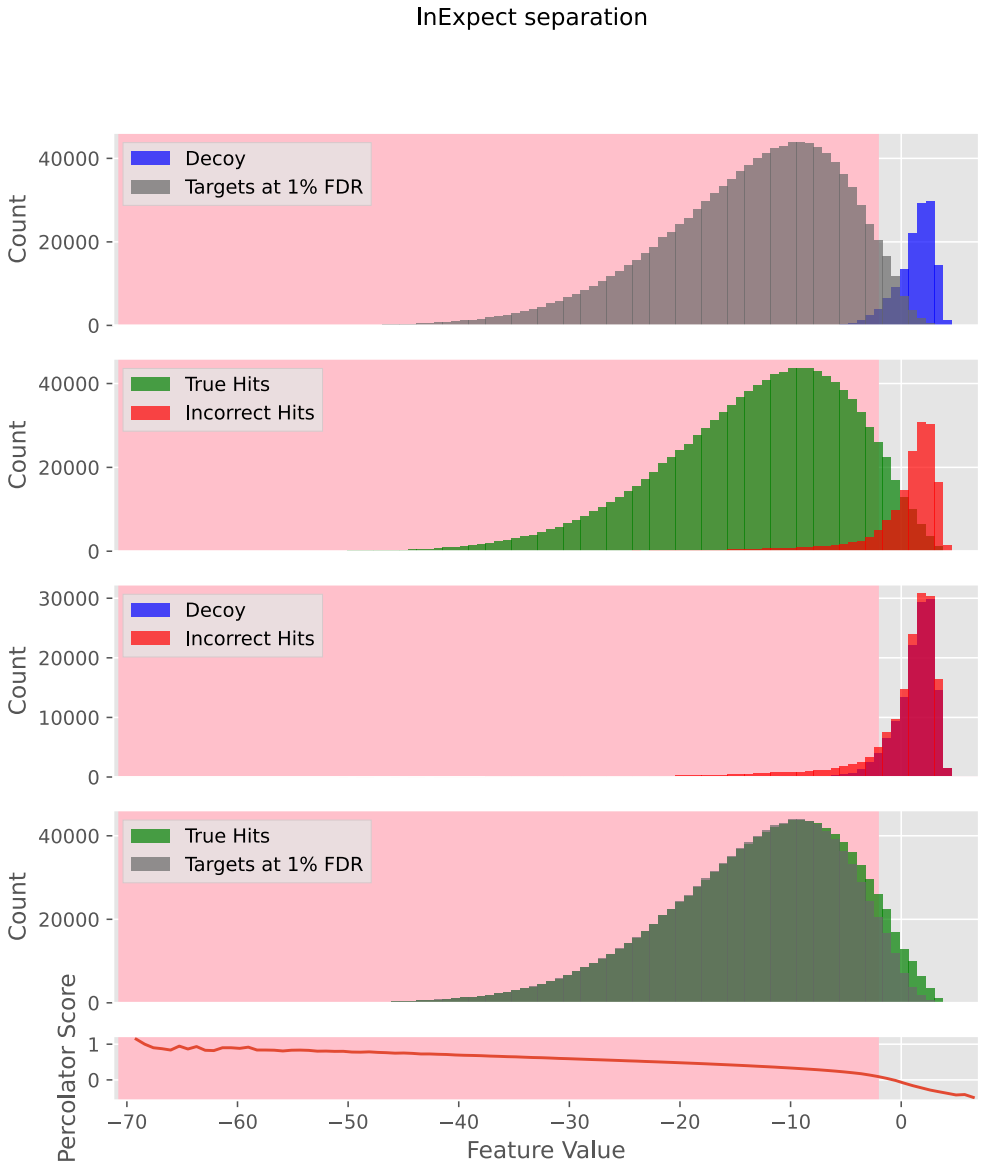
Supplementary Figure 3 – Distribution skew in ProteomeTools search results using Comet along the difference between best and second-best score (deltCn) feature.



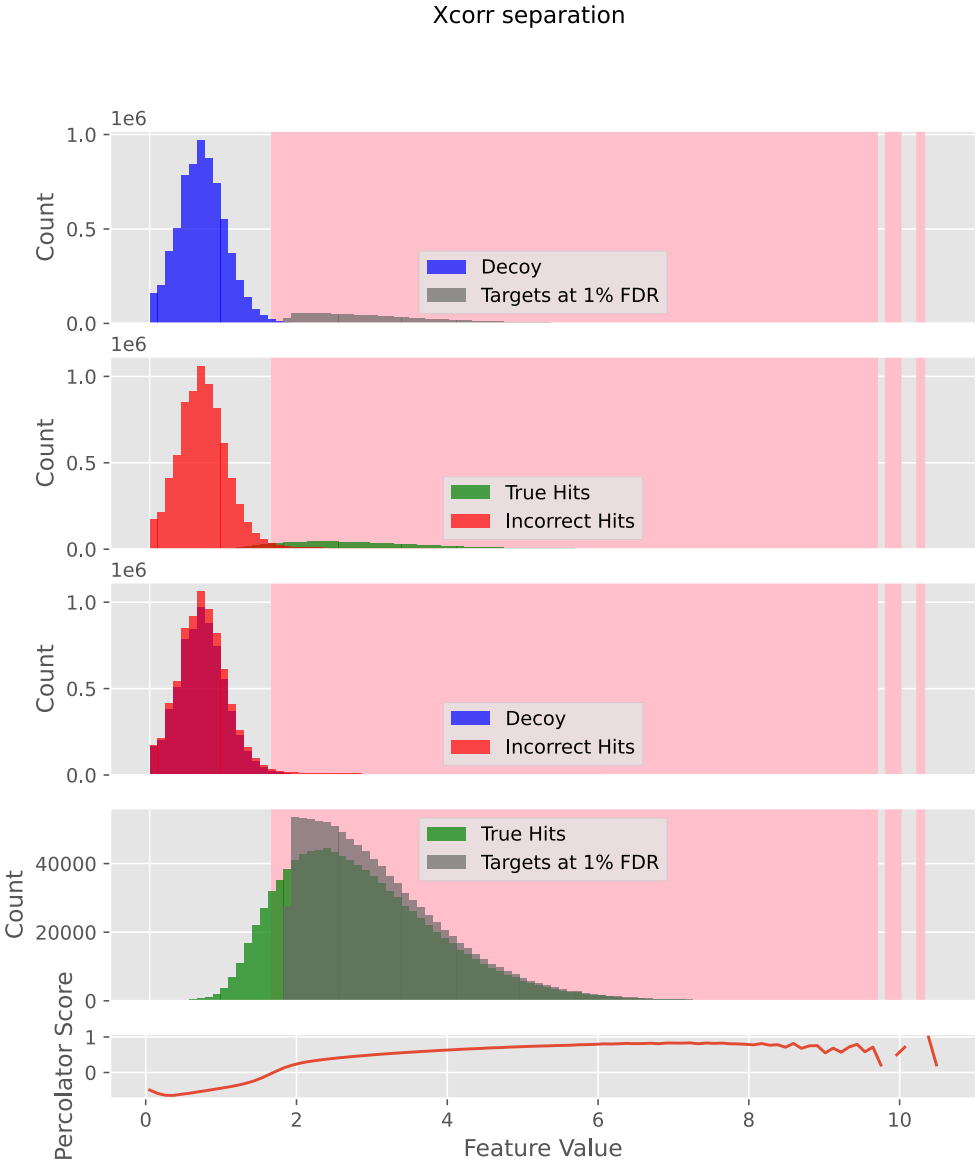
Supplementary Figure 4 – Distribution skew in ProteomeTools search results using Comet along the difference between best and worst score (deltLCn) feature.



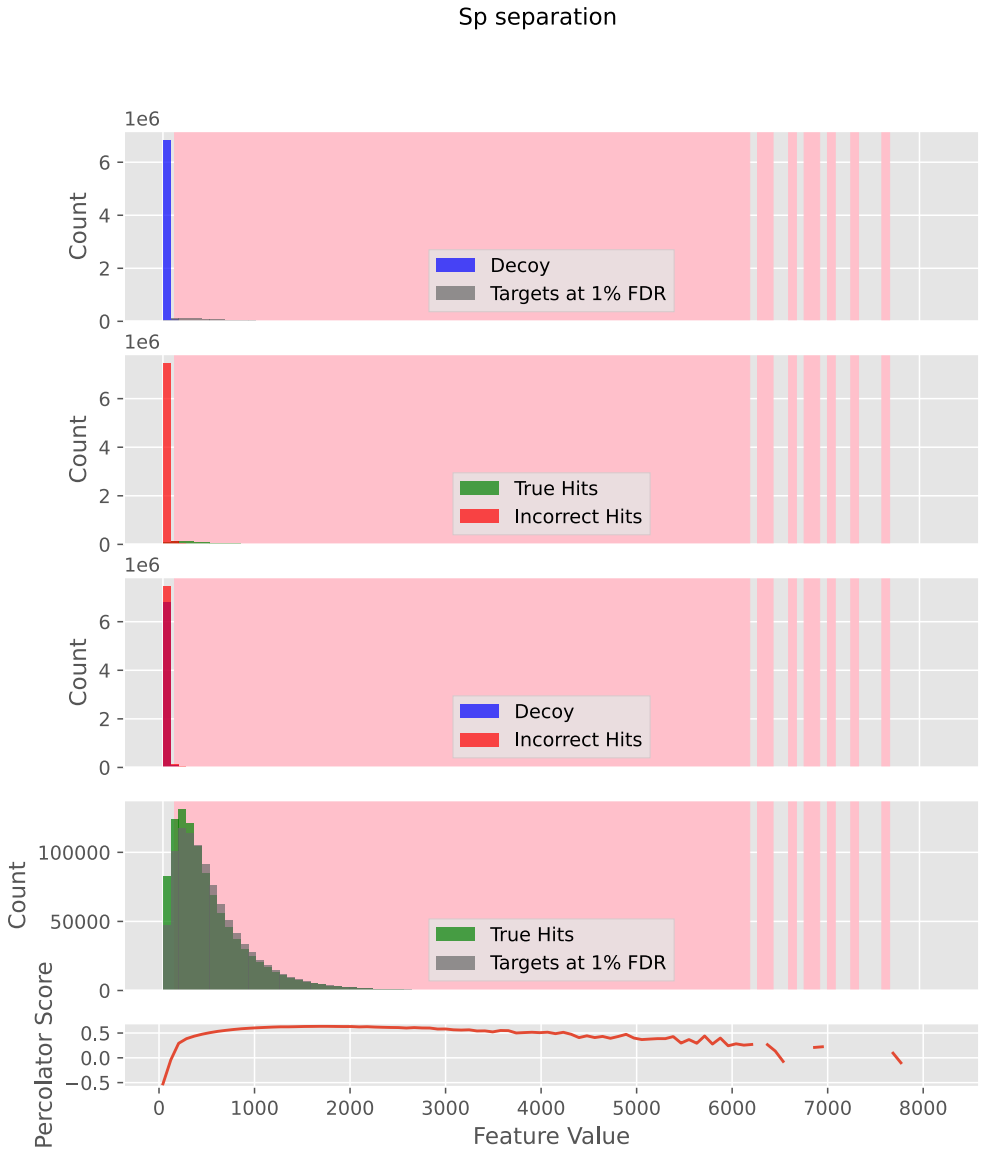
Supplementary Figure 5 – Distribution skew in ProteomeTools search results using Comet along the ion intensity fraction (IonFrac) feature.



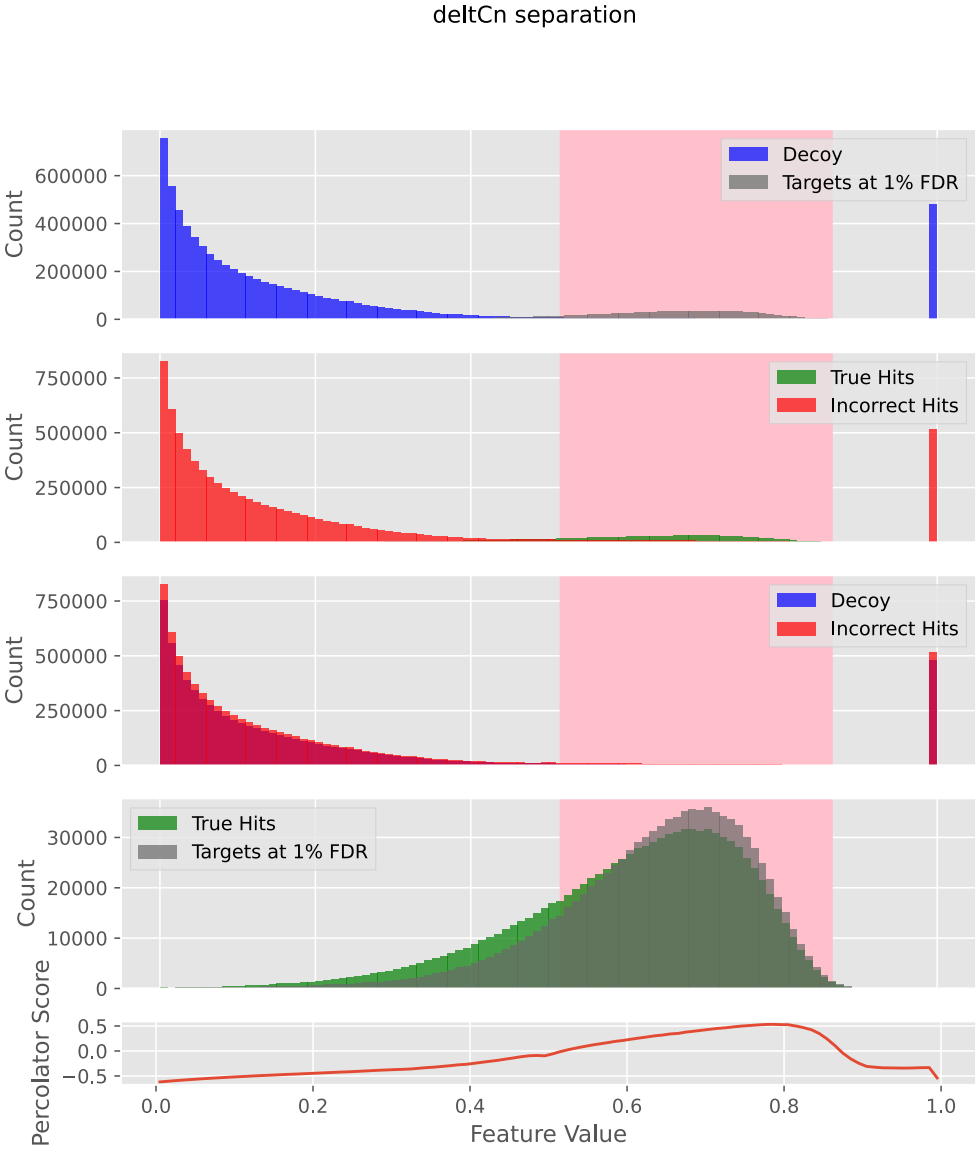
Supplementary Figure 6 – Distribution skew in ProteomeTools search results using Comet along the natural logarithm of the expectation (lnExpect) feature.



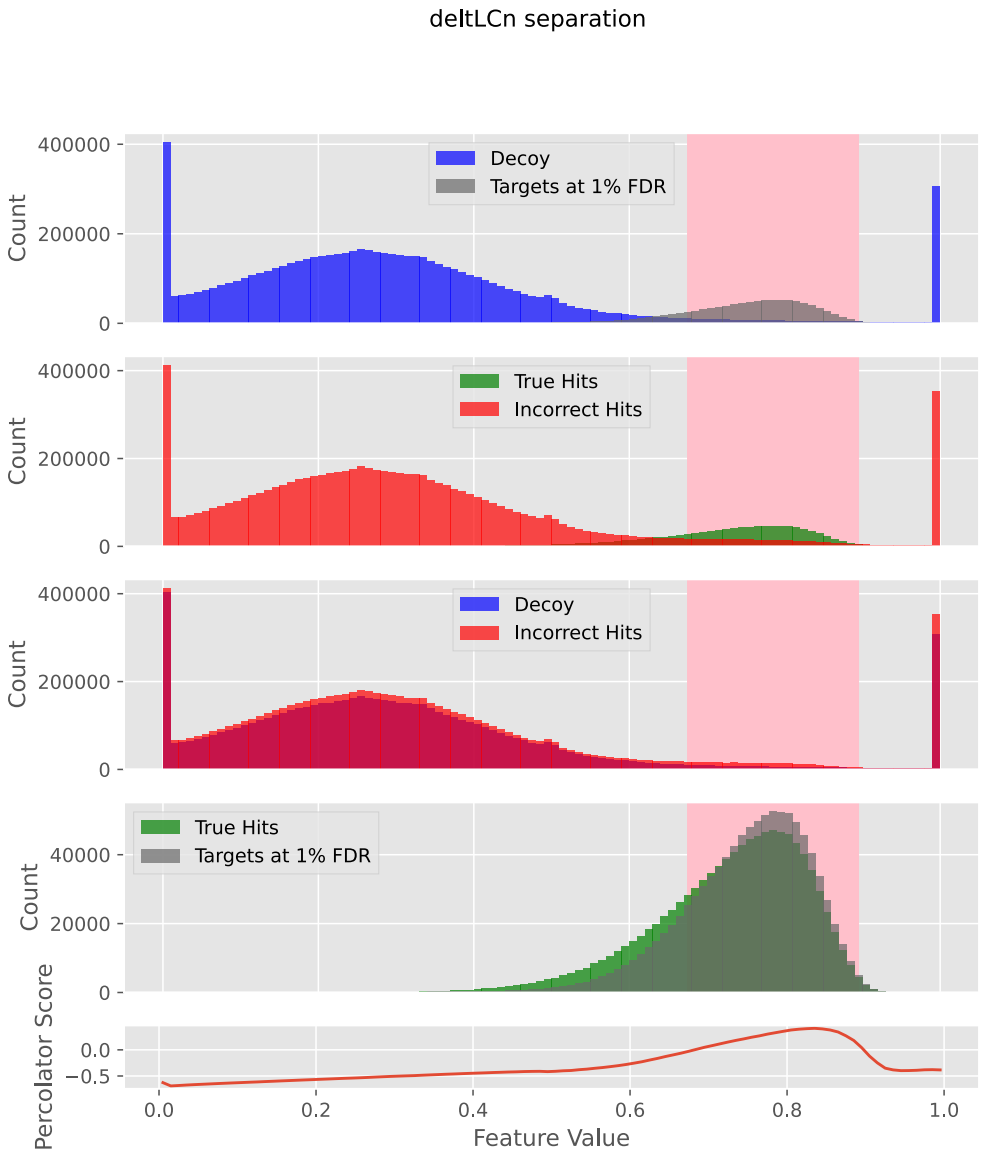
Supplementary Figure 7 – Distribution skew in Massive-KB search results using Comet along the Xcorr feature.



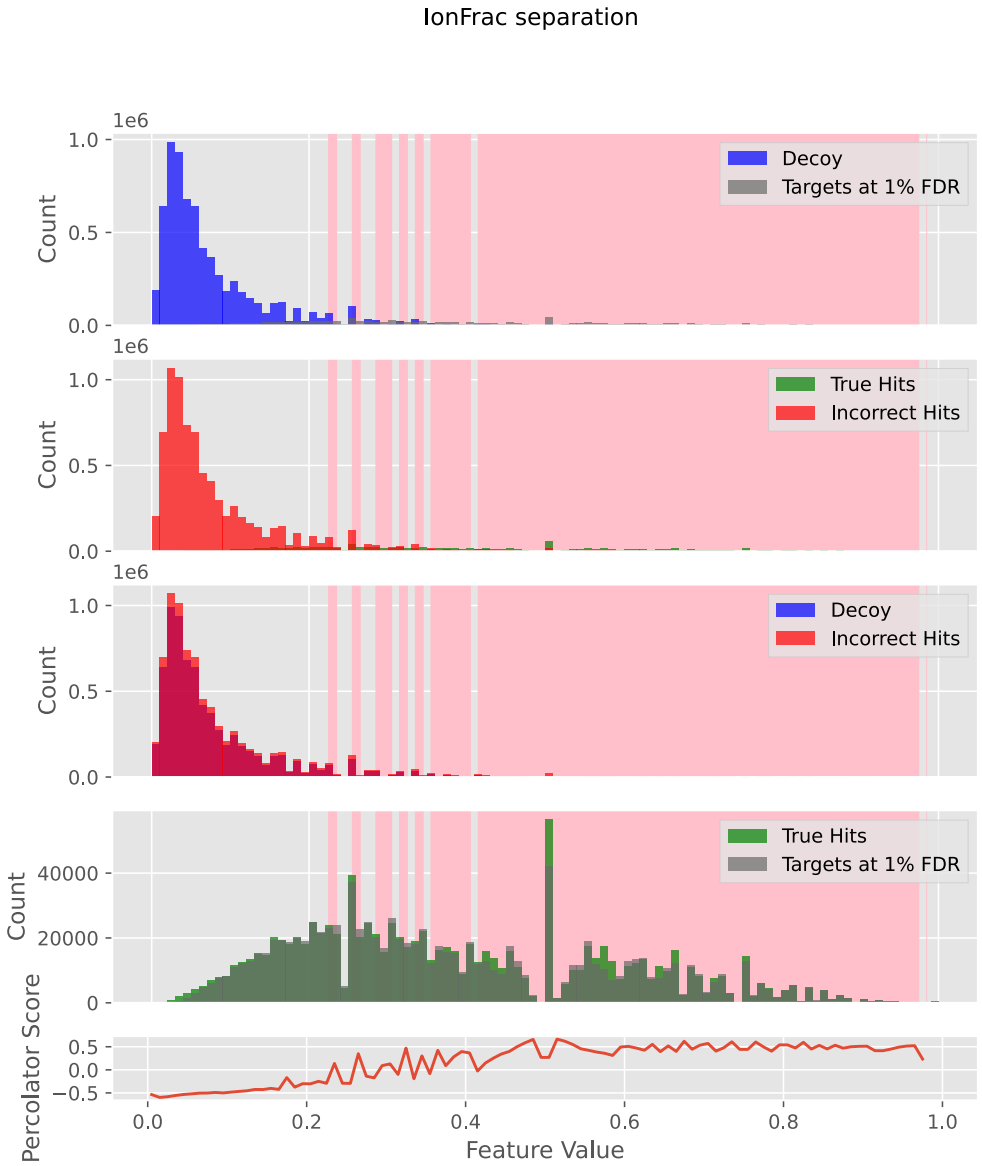
Supplementary Figure 8 – Distribution skew in Massive-KB search results using Comet along the Sp feature.



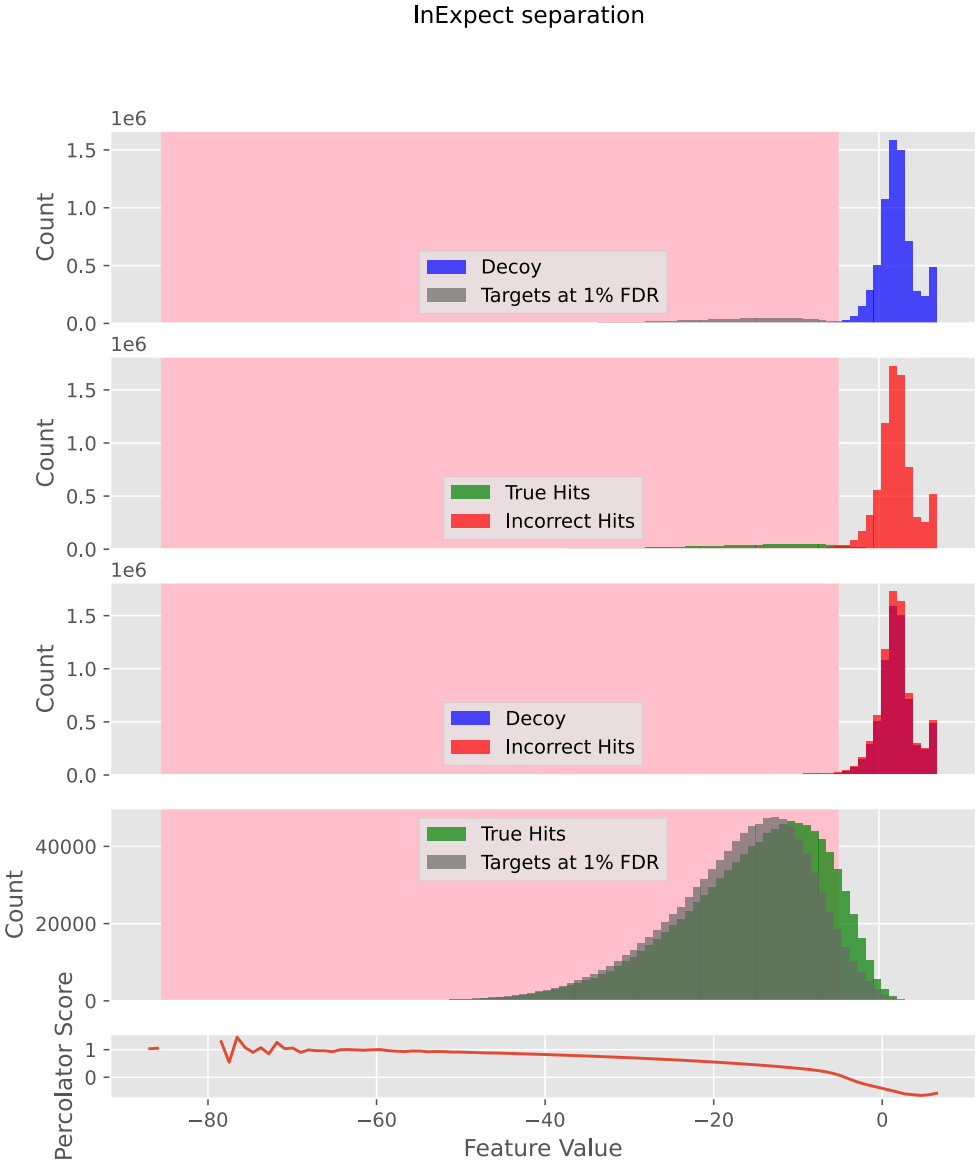
Supplementary Figure 9 – Distribution skew in Massive-KB search results using Comet along the difference between best and second-best score (deltCn) feature.



Supplementary Figure 10 – Distribution skew in Massive-KB search results using Comet along the difference between best and worst score (deltLCn) feature.

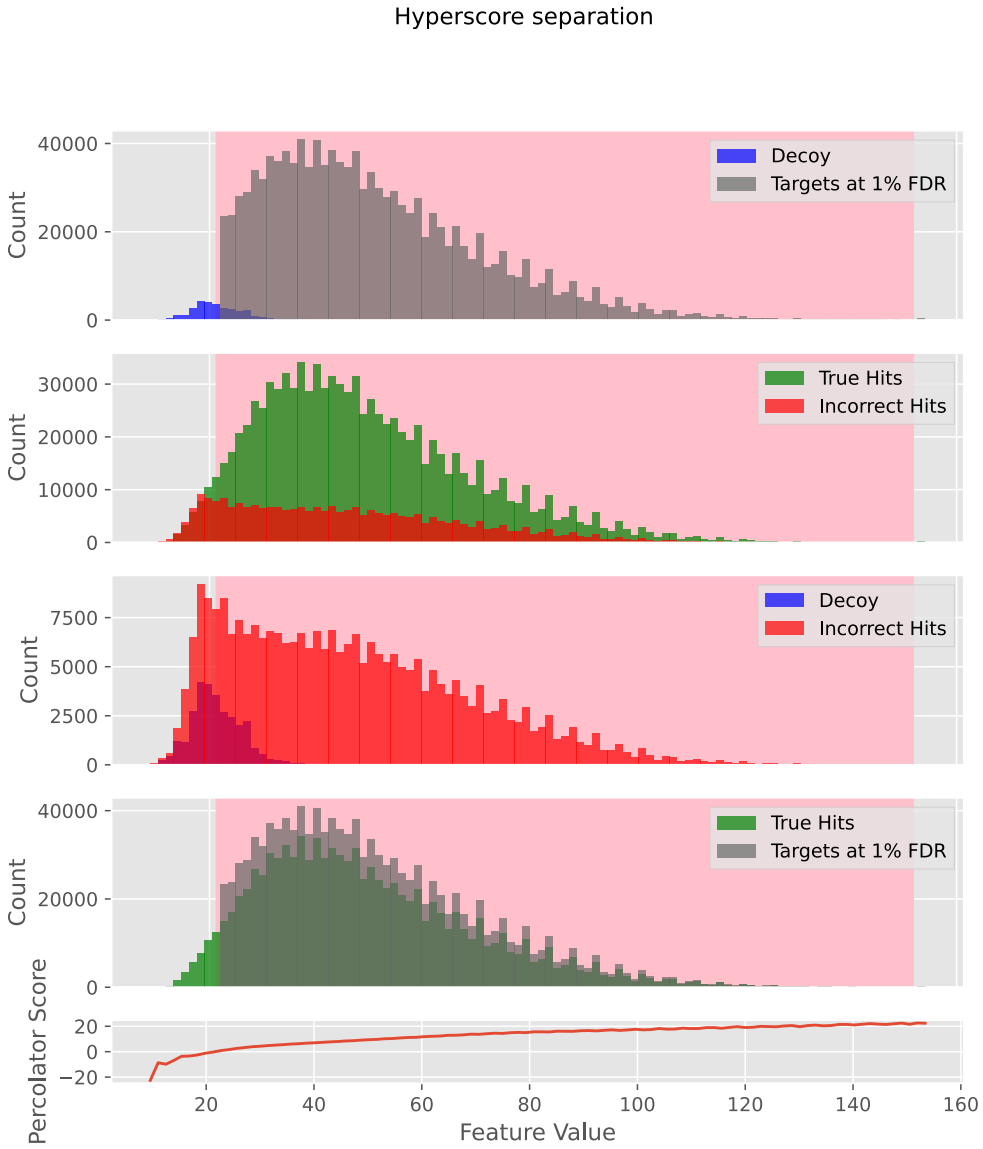


Supplementary Figure 11 – Distribution skew in Massive-KB search results using Comet along the ion intensity fraction (IonFrac) feature.

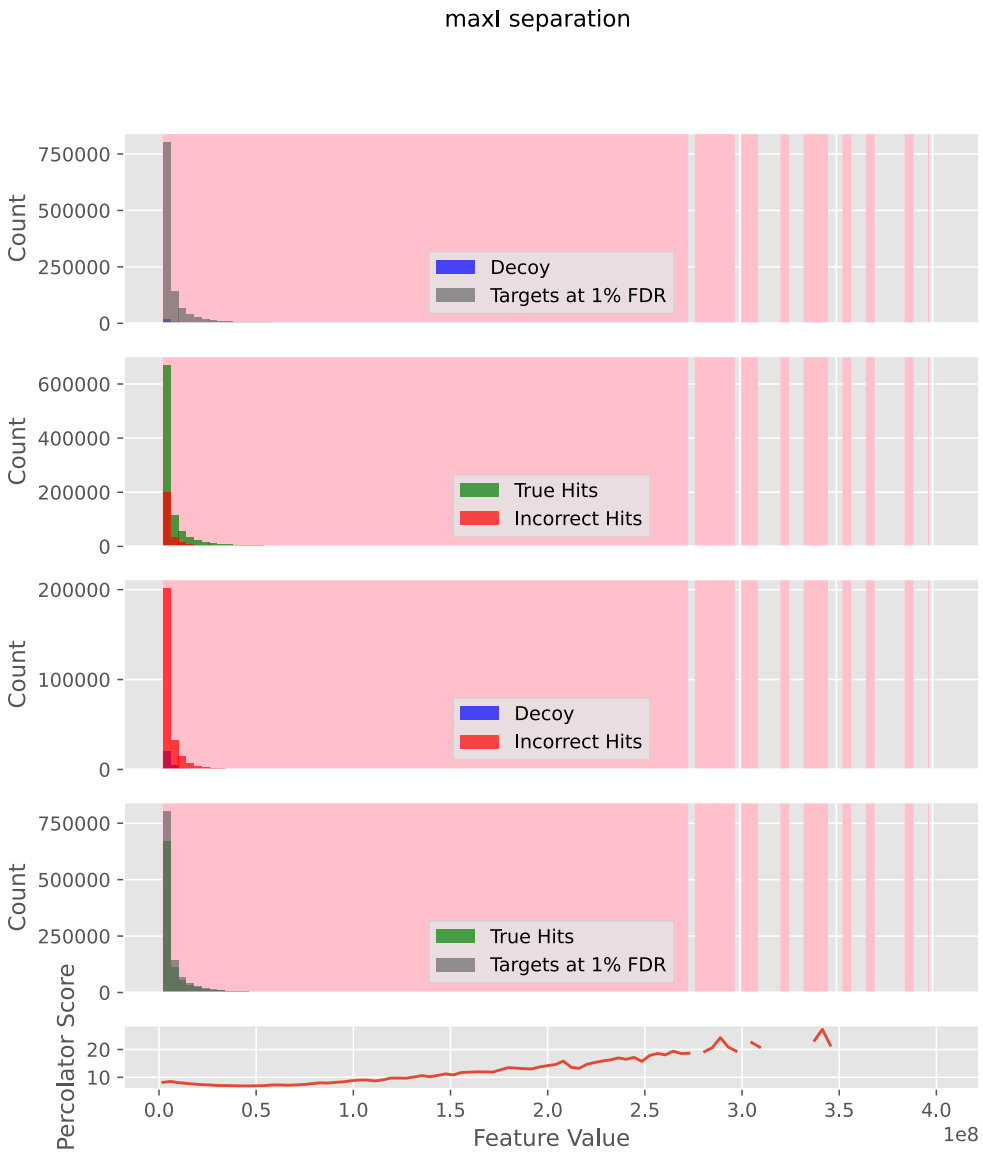


Supplementary Figure 12 – Distribution skew in Massive-KB search results using Comet along the natural logarithm of the expectation (lnExpect) feature.

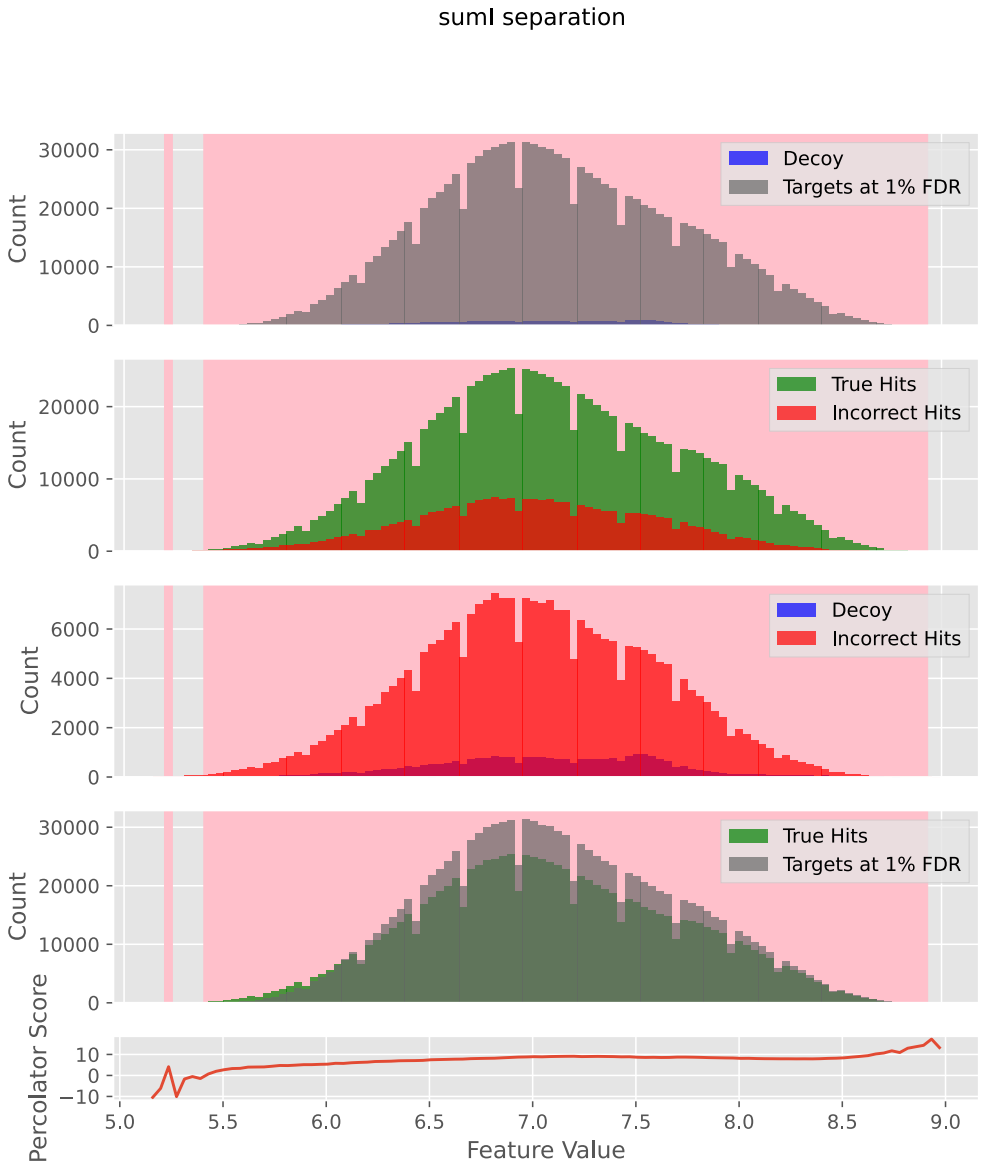
C.1.2. Distribution Skews With X!Tandem



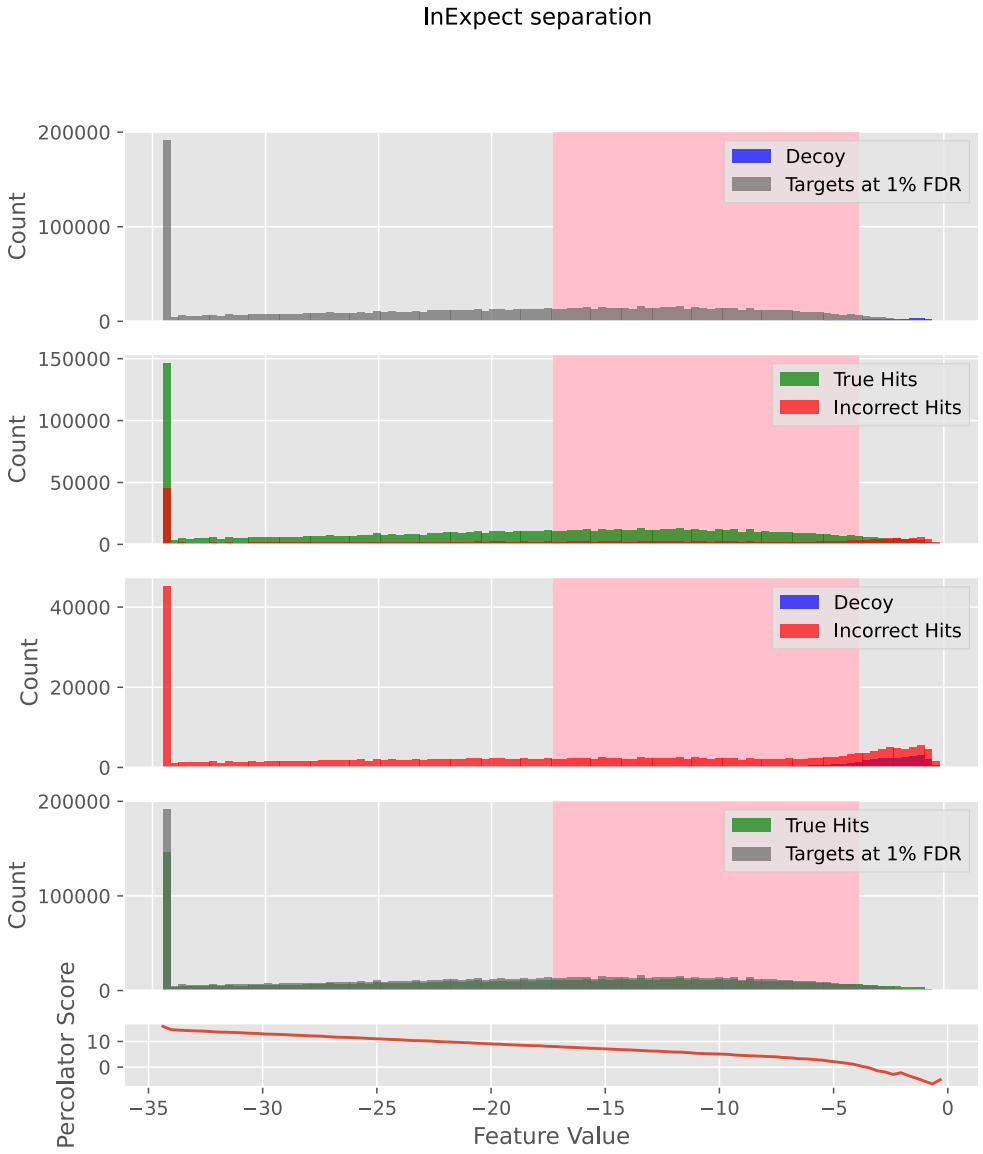
Supplementary Figure 13 – Distribution skew in ProteomeTools search results using X!Tandem along the Hyperscore feature.



Supplementary Figure 14 – Distribution skew in ProteomeTools search results using X!Tandem along the max intensity (maxI) feature.



Supplementary Figure 15 – Distribution skew in ProteomeTools search results using X!Tandem along the sum of intensity of matched ions (sumI) feature.



Supplementary Figure 16 – Distribution skew in ProteomeTools search results using X!Tandem along the natural logarithm of the expectation (lnExpect) feature.