Fachhochschul-Bachelorstudiengang
MEDIZIN- UND BIOINFORMATIK
A-4232 Hagenberg, Austria

Comparative Work on De Novo Mass Spectrometry Algorithms

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science in Engineering

Eingereicht von

Lukas Steininger

Begutachtet von FH-Prof. DI (FH) Dr.techn. Viktoria Dorfer, MSc

Hagenberg, Mai 2024

Declaration

Contents

Abstract

In the field of proteomics, de novo sequencing describes the identification of peptides solely based on the mass spectrum. Compared to the current standard of database search, no reference genome is used and identification of novel peptides is possible. However, the accuracy with which de novo sequencing tools predict sequences is usually lower than the one of database searches. In this work the four de novo algorithms *DeepNovo*, *PEAKS*, *Novor* and *DirecTag* are compared to each other and to database search results provided by the dataset. Furthermore, the best performing algorithm will be named. The spectral data analysed in the scope of this analysis is publicly available. Predictions made by the algorithms are linked to actual sequences of peptides that were included in the mass spectrometry experiments. Predicted and actual sequences are scored by applying multiple similarity metrics. Percent identity and percent similarity are used to make claims about the identification performance of each algorithm. Alignment scores and Levenshtein distance give insight in the quality of the predictions. In addition, the number of predictions and the number of correctly identified sequences will be incorporated in the analysis. Based on the results, it is evident that de novo sequencing performance is below that of database searches. However, the produced data allows for the formulation of a definitive best performing algorithm. Novor performed considerably better than DeepNovo and DirecTag. Although PEAKS was able to match Novor's performance, it only provided predictions for around 60% of all spectra, whereas Novor covered 99% of all spectra. The code used for this analysis is available in the projects GitHub repository [46].

Kurzfassung

Im Bereich der Proteomik beschreibt die De-Novo-Sequenzierung die Identifizierung von Peptiden allein auf der Grundlage des Massenspektrums. Im Vergleich zum derzeitigen Standard der Datenbanksuche wird kein Referenzgenom verwendet und die Identifizierung neuartiger Peptide ist möglich. Allerdings ist die Genauigkeit, mit der De-novo-Sequenzierungswerkzeuge Sequenzen vorhersagen, in der Regel geringer als bei der Datenbanksuche. In dieser Arbeit werden die vier De-novo-Algorithmen *DeepNovo*, *PEAKS*, *Novor* und *DirecTag* miteinander und mit den Ergebnissen der Datenbanksuche aus dem Datensatz verglichen. Außerdem wird der Algorithmus mit der besten Leistung benannt. Die Spektraldaten die im Rahmen dieser Analyse analysiert wurden, sind öffentlich zugänglich. Die von den Algorithmen erstellten Vorhersagen sind mit den tatsächlichen Peptidsequenzen verknüpft, die in den massenspektrometrischen Experimenten enthalten waren. Die vorhergesagten und die tatsächlichen Sequenzen werden durch Anwendung mehrerer Ähnlichkeitsmetriken bewertet. Prozentuale Identität und prozentuale Ähnlichkeit werden verwendet, um Aussagen über die Identifizierungsleistung der einzelnen Algorithmen zu treffen. Alignment Scores und Levenshtein-Distanz geben Aufschluss über die Qualität der Vorhersagen. Darüber hinaus werden die Anzahl der Vorhersagen und die Anzahl der korrekt identifizierten Sequenzen in die Analyse einbezogen. Anhand der Ergebnisse wird deutlich, dass die Leistung der De-novo-Sequenzierung unter der von Datenbanksuchen liegt. Die gewonnenen Daten ermöglichen jedoch die Formulierung eines endgültigen Bestleistungsalgorithmus. Novor schnitt deutlich besser ab als DeepNovo und DirecTag. PEAKS konnte zwar mit der Leistung von Novor mithalten, lieferte aber nur Vorhersagen für etwa 60% aller Spektren, während Novor 99% aller Spektren abdeckte. Der für diese Analyse verwendete Code ist im GitHub-Repository des Projekts verfügbar [46].

Chapter 1

Introduction

1.1 Motivation

In the field of proteomics, two major approaches regarding the identification of peptides given mass spectrometry data are database search and de novo sequencing. Both have their advantages and disadvantages. Database searches have a higher accuracy than de novo sequencing. A major drawback, however, is the fact that a genome for the species or a close relative is needed. De novo sequencing does not rely on any collection of references, since it infers the sequence based only on the mass spectrometry data. Thus, it can be used on source material from any species. Despite that, the accuracy of these predictions is very low compared to results of database searches. [1]

1.2 Objective and Approach

The objective of this work is to compare four selected de novo sequencing algorithms to each other and to results of a database search. Results of the comparative analysis will give grounds to determine the best performing algorithm. The four algorithms will be used with their default parameters, including pretrained models for any deep learning approaches. Settings regarding the mass spectrometry data (fragmentation type, tolerance, etc.) will be changed to ensure proper handling of the data. All programs are used on three different datasets to assess their reproducibility. The datasets come with results of a database search and a list of peptides that were included in the original mass spectrometry experiment. Predictions made by the algorithms are first checked against the database search result. Predictions without a match are checked against the inclusion list. The primary usage of the database search results is due to the otherwise high computational effort. Predicted and actual sequences are scored with multiple metrics to give insight into the identification as well as the quality of the prediction. Moreover, the number of predictions and identified sequences will be incorporated into the analysis as well.

Chapter 2

Background

2.1 Mass Spectrometry

Mass Spectrometry (MS) has a multitude of application areas. It can be used for qualitative and quantitative analysis alike. Therefore, MS enables researchers to verify whether an analyte is present at all, as well as to make assumptions of how much of the analyte is present. [2] In the field of proteomics, Mass Spectrometry is used, among other things, to identify peptides and proteins and study protein-protein interactions. [3]

In all its steps, MS encompasses a multitude of techniques, each with certain advantages and disadvantages. For that reason, no single perfect setup exists that works for every type of experiment. [3] Choosing the techniques should be dependent on the samples and the goals that want to be reached. [3]

2.1.1 Structure of a Mass Spectrometer

A Mass Spectrometer consists of three parts: an ion source, a mass analyser and a detector. [4] The ion source takes molecules of the sample and ionises them. Each ion has a positive charge. Then, the mass analyser filters ions based on their mass-to-charge ratio (m/z). This ratio sets the mass of the ion in relation to the number of charges it has. Finally, a detector measures the ions and converts the data into a mass spectrum chart. [4]

Ion Source

Two commonly used techniques to ionise peptides are Electrospray Ionization (ESI) and Matrix-assisted Laser Desorption/Ionisation (MALDI). [4] MALDI ionises the analyte, which is contained in a crystalline matrix. With ESI, however, the analyte is suspended in a solution. This allows the analysis of more complex analytes, since ESI can be paired with separation tools like liquid chromatography. [4]

Mass Analyser

The information in the following paragraphs is summarized from El-Aneed and co-workers [5]. In an MS experiment, the ions produced by the ion source are usually filtered based on their mass-to-charge ratio. This process can happen either electrically or via magnetic fields. Two important mass analysers are the quadrupole analyser (Q) and the time-of-flight analyser (ToF).

The quadrupole analyser consists of four electrical rods. Two of them receive a potential (U) via direct current, the other two receive a potential (V) via an alternating radio-frequency (frequency ω). Positively charged ions are drawn to the negative pole of the latter two rods. Since the electrical polarity is switched continuously, ions oscillate while traversing the mass analyser. By fine-tuning the parameters U, V and ω, only ions within a certain tolerance of the desired m/z will be able to fully traverse the analyser and be detected by the detector. Ions that do not possess this certain trajectory will eventually crash into the poles.

The time-of-flight analyser does not remove ions that do not fit the desired mass-to-charge ratio. It rather makes use of the time an ion needs to get to the detector. Ions follow a linear path through the mass spectrometer. They are accelerated in the first part of the machine and reach their peak velocity in a second part. The m/z value can be calculated with the time-of-flight t_F, the voltage E, the length of the acceleration region s and the length of the free flight region x.

Detector

After passing the mass analyser, the m/z value of the remaining ions is measured. [6] Different technologies have been developed to quantify the m/z value, one of the most used ones is the Electron Multiplier. [7]

The Electron Multiplier facilitates a principle called secondary electron emission. The initial ions that passed the mass analyser strike the first of multiple so-called dynodes. Contact of an ion or electron with these dynodes releases additional electrons. These electrons are attracted to the next dynode, where more electrons are emitted. Usually this step is repeated until an amplification of the factor 10^6 has been reached. [6]

2.1.2 Tandem Mass Spectrometry

In Tandem Mass Spectrometry (MS/MS) two mass analysers are used in order to further inspect a single mass-to-charge ratio. This is achieved by the additional usage of a collision cell and a second mass analyser. The rest of the experimental setup, i.e., the ion source and detector, remain the same. [5]

Ions that are filtered by the first mass analyser (precursor ions) are led into the collision cell where they collide with atoms of an inert gas (e.g., helium, argon, etc.). [5] By colliding, kinetic energy is converted into internal energy and the ion fragments at

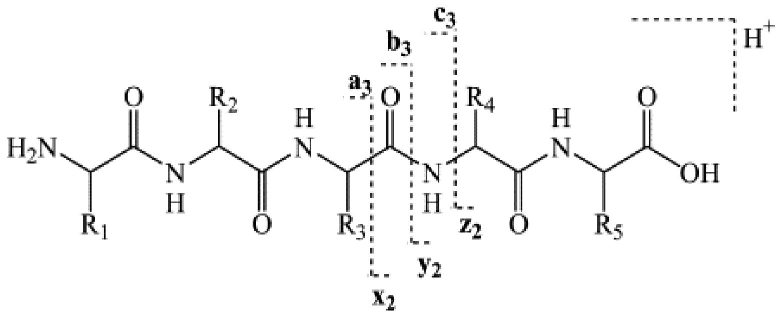

Figure 2.1: Sketch of fragment ion types from [8], marking the fragmentation position in the peptide backbone for ion types a, b, c, x, y, z. The subscripts in the ion pair a_3 and x_2 correspond to the 3 residues before and 2 residues after the fragmentation site.

unstable chemical bonds. [8] This process is called Collision-Induced-Dissociation (CID). After that, the resulting fragments (product ions) are again filtered by the second mass analyser. [5] Other methods like Electron Transfer Dissociation (ETD), use radicals to induce dissociation. This approach can be used to fragment the peptide a the site of a Post-Translational Modification (PTM). [9]

Wysocki and co-workers [8] described the types of fragment ions and their terminology, the following information is a summary of their work. Different types of ions can be produced, depending on where in the peptide the fragmentation happens. A nomenclature has been devised to characterize the fragment ion types. Fragmentation always produces a complementary pair of ions. Every ion is marked with a subscript denoting the number of residue side chains to either end of the peptide. Summation of the subscripts of one pair results in the number of residues of the peptide. The ion pairs to occur along the peptide backbone are: a & x, b & y and c & z. Figure 2.1 shows the fragmentation sites for these three pairs.

2.1.3 Typical Use Case

In the field of proteomics, researchers usually facilitate the bottom-up approach (as opposed to top-down). [10] The proteins of interest are extracted from cells by various methods. [4] Since analysis of complete proteins shows little sensitivity, they are sliced into smaller pieces, peptides, by using enzymes. These macro-molecules cut an amino-acid sequence based on a given set of rules in a process called proteolytic digestion. Usually, the enzyme trypsin is used. [10] Complex peptide mixtures are commonly separated via High Pressure Liquid Chromatography (HPLC) and ionised with the ESI method. Identification of the individual peptides, and consequently the proteins, is achieved by performing tandem mass spectrometry. [4]

2.1.4 Produced Data

Information about the mass spectrum is summarized from the work of Nicolescu and co-workers [11]. The product of the analysis is called mass spectrum. The x-axis represents the measured m/z values. The normalised y-axis encodes the intensity or the

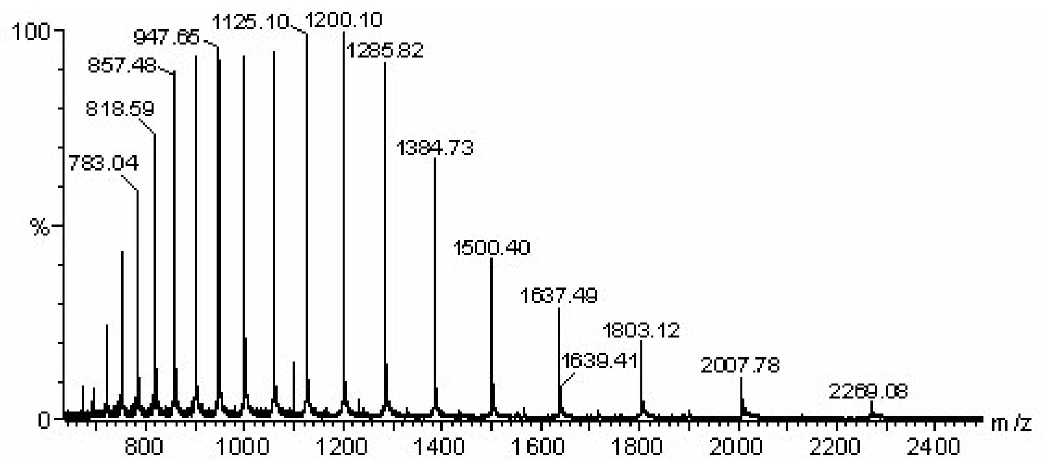

Figure 2.2: Typical mass spectrum from [51]. The hints on the peaks correspond to their m/z on the x-axis and not their abundance.

relative abundance of ions with this specific m/z value. Relative abundance is calculated by finding the most abundant ion (100%) and assigning every other ion their respective percentage value. A typical representation of a mass spectrum can be seen in Figure 2.2.

The following details regarding data formats are compiled from the work of Deutsch [12]. The information of the mass spectrum is written to output files for further usage and analysis. MS-vendors usually encode the read-data in binary files, which are difficult to interpret. New formats facilitating the structured XML syntax have been developed (e.g., mzXML). Additionally, for MS/MS, it is still common to convert the binary files to simple text files containing the reads (e.g., MGF).

2.2 Artificial Intelligence and Dynamic Programming

2.2.1 Neural Networks

Neural networks try to mimic the human brain function. Like real neurons, a multitude of artificial neurons (processing units) are connected with each other. All inputs are summed up, the resulting value is termed activation. If the activation crosses a specific threshold, the neuron in question produces a value, usually between 0 and 1. Connections are subjected to weights, which could increase or weaken the input signal. [13] A classic feed-forward network consist of multiple layers, with each layer having multiple neurons. An input layer receives information (features) and passes the values to a hidden layer. Multiple hidden layers determine how the input relates to the expected output. The processed information is handed to an output layer that presents the result. [14]. The initially random weights of these networks are constantly adjusted. This process is called learning and is achieved by the backpropagation algorithm. [15]

Convolutional Neural Networks (CNNs)

A Convolutional Neural Network (CNN) is based on the above description of a feed-forward network. CNNs are best known for image recognition. The architecture features four different types of layers: input, convolutional, pooling and fully connected. [16] Convolutional layers extract features from the data. Each neuron is only connected to a subset of previous neurons. These connections have weights which are learned in the training process. The set of connections of a neuron are called a filter or kernel. Multiple filters are convolved over the input data and recognise certain patterns. The output of these convolutions are called activation map or feature map.[16] Pooling layers reduce the dimension by selecting a single value of the activation map. Common pooling methods are selecting the maximum or the average value of an $N \times N$ section of the activation map. [17] Convolutional and pooling layers are typically stacked multiple times and end up in a fully connected layer. All neurons in this layer are connected to all neurons in the layer before and after. The fully connected layer connects to the output layer. [16]

Recurrent Neural Networks (RNNs)

A Recurrent Neural Network (RNN) is used to learn patterns from sequential data. Like neural network, RNNs have multiple layers, two of them being the input and output layer. The input consists of multiple vectors, with each vector representing the features at a certain point in time. Unlike neural networks, hidden layers in an RNN create a directed cycle which serves as a memory. [18] The output of a hidden layer is fed back into the same hidden layer. [19] Weights of the connections are trained by an adapted backpropagation algorithm, termed backpropagation through time (BPTT). [18] In the process, the RNN is effectively converted to a feed-forward network. When dealing with long sequences, this can potentially lead to the problem of a vanishing or exploding gradient. [19]

Building upon that problem, the Long Short-Term Memory (LSTM) was developed.

Information (features) is handled by gated cells. LSTMs use an input gate, an output gate and a forget gate. [18] Cells are chained and hold a certain state. One cell takes in the cell state of the previous cell ($c(t-1)$), the hidden state of the previous cell ($h(t-1)$) and the input at a specific time (x_t). The cell state is updated based on the gates. The forget-gate defines which parts of the cell state are kept, the input-gate decides what new information is added to the cell state and the output-gate decides on the output based on the current cell state. [20]

2.2.2 Decision Trees

Decision trees are a supervised machine learning technique that can be used for classification and regression alike. Features can be discrete or continues. [21] A tree consists of nodes and branches. The root node marks the beginning of the tree. Nodes that are both children of other nodes and are themselves parents to other nodes are termed internal nodes. Nodes without any children are leaf nodes. [22] In the case of a classification problem, a decision tree tries to classify the input data by asking a series of questions about the features. To be able to do this, labels (i.e., categories) have to be provided for the data. All nodes, except leaf nodes, are associated with a question that affects the direction of the questioning. If a leaf node is reached in the process, the data point is classified based on the category assigned to the leaf node. [21]

The aim, in asking questions, is to split a list of data points so that the splits are as homogeneous as possible regarding their label. Some metrics that describe the impurity are the entropy and the Gini index. A tree is constructed by repeatedly choosing questions that minimize the entropy (or the Gini index). [21] A tree is expanded until certain criteria are met. Alternatively, the tree can be grown to full size and then pruned down to the optimal size by removing nodes with little information gain. [22]

2.2.3 Dynamic Programming

The following information is a summary extracted from the work of Buhisi and coworkers [23]. Dynamic programming is a concept used in computer science that solves problems by breaking them down. However, the problem has to have optimal substructure and sub-problems need to overlap. This means that a problem can be solved by dividing it into smaller problems. Overlapping of sub-problems means that a specific problem is solved once and the solution is stored. Should it come up again, the stored result is used instead of solving it again. Two approaches are common for dynamic programming: top-down and bottom-up. In the top-down method, a problem is repeatedly broken down, solved and the solution is stored. The bottom-up method solves every sub-problem beforehand and stores the solutions.

Chapter 3

State of the Art De Novo Algorithms

3.1 DeepNovo

This approach for de novo peptide sequencing is described in [24] and merges two major techniques: neural networks and dynamic programming. Two neural networks facilitate learning from the input data and predict amino acid classes, while dynamic programming is used to filter for amino acids that fulfil a mass constraint. By combining these methods, DeepNovo was able to achieve better results than the current standards at the time of publication (2017; i.e., Peaks [25] (chapter 3.2), Novor [26] (chapter 3.3), PepNovo [27]). Performance was measured on a broad test set. Resolutions as well as used MS techniques vary. This ensures fair evaluation [24].

3.1.1 Key Concepts

The following descriptions are a summary extracted from the work of Tran and co-workers [24]. DeepNovo can be split into two major steps: feature construction via the input data and prediction of symbol. A symbol represents one of the 20 amino acids, start, stop, padding or one of three modifications, totalling to 26 possible symbols. The prediction step is repeated until the system produces the symbol 'stop'. The way how feature construction and iterative prediction are combined is shown in Figure 3.1.

Feature Construction

Predicting a peptide requires a tandem mass spectrum in the MGF format, which is transformed into an intensity vector. Each value at an index corresponds to the intensity at a certain m/z-ratio. Two types of resolution are available: low (0.1 Da) and high (0.01 Da). The amino acid 'Alanine' has a mass of 71.0 Da and would therefore have a value at the indices 710 and 7100 at low and high resolutions respectively. A maximum m/z-value is assumed (e.g., 5,000 Da), which in combination with the resolution allows for the calculation of the vector length.

In order to recognise patterns (i.e., amino acids) in the data, the intensity vector is further processed. A copy of the intensity vector is cut at the index corresponding to the m/z-ratio of an amino acid. The snipped copy is filled with the value 0 in order to gain

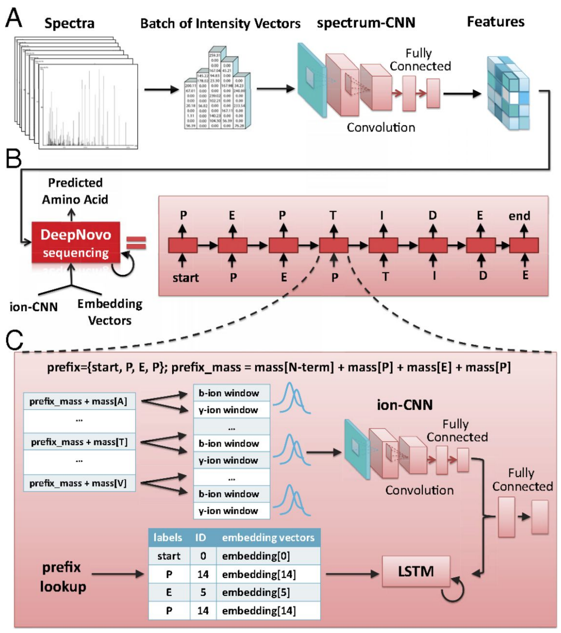

Figure 3.1: DeepNovo model from [24]. **(A)** Conversion of input spectra to intensity vector for use in spectrum-CNN. This output is used to initialize the LSTM. **(B)** The iterative process of predicting one amino acid per iteration, uses the ion-CNN and the LSTM. Prediction is stopped as soon as the symbol “stop” is produced. **(C)** Details of a single prediction step. Both the ion-CNN and the LSTM predict the next amino acid based on the ones that were predicted before. Their outputs are combined.

equal size. The original and the altered vector are joined in the second dimension. This step is repeated for all 26 symbols and leads to a result of the dimensions $1 \times 2 \times N \times 26$ with N being the maximum m/z-value.

A batch of 128 of the altered intensity vectors is used as input data for the spectrum-CNN, so patterns can be recognized and learned. Two convolutional layers of the sizes $2 \times 10 \times 26 \times 32$ and $1 \times 5 \times 32 \times 64$ are chained, and their result is used as input to a fully connected layer of 512 neurons. The output of the CNN, a vector of length 512, is utilized as a source for the LSTM, another neural network, used to learn the emerged patterns.

Prediction of Amino Acids

Predicting the whole peptide sequence is achieved by predicting one symbol at a time. This is done by combining the power of two neural networks. On the one hand, a convolutional network, termed ion-CNN, learns features from processed data. On the other hand, an RNN of type LSTM takes the previous predictions into account to deliver information used in the prediction. Both their outputs are combined to predict the next symbol and are further used as the input for the next iteration.

The ion-CNN makes use of the prefix. The prefix is a sequence with all the predicted symbols so far. By adding the mass of the N-terminus and the amino acids, the prefix mass is calculated. After that, for each possible combination of prefix and symbol, eight different kinds of ions are calculated (b-ions, y-ions, six others). These masses, including a tolerance of 1.0 Da, are checked against the intensity vector. The values are collected into an array of the size $26 \times 8 \times 10$ with 10 being the 1.0 Da tolerance at a low resolution of 0.1 Da.

Batches of size 128 of these arrays are used as input of the ion-CNN. This convolutional network facilitates two convolutional layers in order to learn local features. Two layers are deemed enough since the bell shape of the features (amino acid peaks) is trivial to learn. Furthermore, a variety of filters is used, to accommodate the different amino acids. Convolution results are fed into a fully connected layer of 512 neurons. The final layer consists of 26 fully connected neurons, each one corresponding to one of the 26 symbols.

An LSTM predicts the next symbol by taking the previous predicted symbols into account. The 26 possible symbols are encoded in embedding vectors of size 512. Stacking these 26 vectors creates the embedding array of size 512×26. Predicting the symbol at position t is composed of the following steps: (1) using the output of the spectrum CNN to initialize the LSTM, (2) the LSTM iterates $i = t - 1$ times, input for each iteration is the embedding vector row that corresponds to the predicted symbol at position i, (3) the symbol at position t is predicted by using the output of the LSTM at iteration $t - 1$ and thus is encoded as vector of size 512.

Linear transformation of combined outputs produces prediction probabilities for amino acid classes. The output of the LSTM and the second to last layer of the ion CNN are merged into a vector of size 1024. Followed by two fully connected layers of 1024 and 26 neurons, an output vector of size 26 is produced. Values contained in that vector, so called logits, are used to calculate probabilities for each amino acid class, with the most probable class being selected. Amino acids of the selected class are filtered based on their suffix mass via a knapsack algorithm, a dynamic programming approach. This ensures correct peptide masses of the final peptide sequence.

3.1.2 Application of the Algorithm

DeepNovo is developed with the programming language Python, version 2.7. The corresponding GitHub repository [28] provides the source code as well as instructions to use the program. In order to run the algorithm, Python has to be installed. One way to do this is via a virtual Conda environment. Furthermore, the following dependencies need to be loaded: Cython (3.0.10), biopython (1.76), tensorflow (1.2.0) and pyteomics (3.4.1). The author also provides a pretrained model and a matrix with values used for the knapsack algorithm. Training a custom model is not within the scope of this thesis, thus this pretrained model is used for the analysis. Both files have to be put with all the others file to allow proper function.

Since the algorithm uses Cython, an optimized version of Python with performance

Program 3.1: Code to start a DeepNovo prediction from the command line

```
python deepnovo_main.py --train_dir train.dir --decode --beam_search --beam_size 5
```

comparable to C [29], it has to be built first. Input data and other algorithm parameters are specified in the file "deepnovo_config.py". See 4.2 for requirements regarding this file. A prediction can be started with the command seen in Program 3.1. The predicted result can be found in the file "../train.example/decode_output.tab".

3.2 PEAKS

The algorithm behind PEAKS, as described in [25], predicts peptide sequences via a dynamic programming algorithm and provides a confidence score for the sequence as well as the amino acids of the sequence. By this, even when no complete peptide sequence can be found, highly confident sequence tags can be detected. Since its development in 2003, PEAKS is constantly updated and eventually has become a commercial software.[47] Recently, even the approach demonstrated by DeepNovo (3.1) has been incorporated into the platform. [52]

3.2.1 Key Concepts

All details provided below are a summary extracted from the work of Ma and co-workers [25]. The algorithm behind PEAKS works in four stages: (1) preprocessing, (2) computation of candidates, (3) revaluation and (4) confidence scoring.

Preprocessing

PEAKS is able to work with MS/MS spectra in a variety of formats and processes the peaks for an enhanced workflow. While being able to read formats like Mascot Generic Format (.mgf) or Sequest (.dta), the authors of [25] state that the best performances were achieved with the mass spectrometers raw data files. One reason for that might be, that the processing steps taken by manufactures are suboptimal for the use case of de novo sequencing. Additionally, this algorithm incorporates novel approaches for noise filtering and peak centering that are unfortunately not further discussed in the paper.

Computation of Candidates

By computing a reward/penalty for every mass value m, the best 10,000 candidate sequences are found through maximizing this reward via a dynamic programming. For a given precursor mass, every possible combination of amino acids that results in that mass is generated. Subsequently, a reward for every b- and y-ion in every candidate peptide is determined. The reward considers the distance of the theoretical peak closest to an observed peak in the spectrum, the relative abundance of the observed b-/y-ion peak and the relative abundance of other, supporting ion peaks if existing. When there is no observable peak within a tolerance of δ of the theoretical peak, the system uses a negative constant to act as a penalty. A dynamic algorithm is then used to find sequences that maximize the reward, retaining the top 10,000 sequences.

Computation for both, b- and y-ions, varies in small details: (1) the y-ion reward uses the three supporting ion types x, y-H_2O and y-NH_3, while the b-ion reward uses the four supporting ion types a, c, b-H_2O and b-NH_3, (2) the b-ion reward is multiplied by 0.5, so the algorithm favours y-ion since they are more ample in an experimental setting with tryptic digestion. The corresponding equations can be seen in Equation 3.1 and Equation 3.2.

$$R_y = f\left(\frac{h_1}{h}\right) \times f\left(\frac{h_2}{h}\right) \times f\left(\frac{h_3}{h}\right) \times \exp\left(-\left(\frac{m^{'} - m}{\delta}\right)^2\right) \times log(h) \tag{3.1}$$

$$R_b = f\left(\frac{h_4}{h}\right) \times f\left(\frac{h_5}{h}\right) \times f\left(\frac{h_6}{h}\right) \times f\left(\frac{h_7}{h}\right) \times \exp\left(-\left(\frac{m^{'} - m}{\delta}\right)^2\right) \times log(h) \tag{3.2}$$

where:

δ	= mass error
m	= mass of b- or y-ion
$\mathrm{m}^{'}$	= mass of observed peak
h	= relative abundance of observed peak
h_i	= relative abundance of supporting ion types
$f(\frac{h_i}{h})$	= supporting function, if h_i comparable to h returns value >1, else 1

Reevaluation

Top scoring candidates of the previous step are rescored more strictly. Reevaluation is done by reducing the mass error δ in the equations above, as well as introducing two new ion types in addition to the b- and y-ions. Rewards are now also calculated for immonium ions and internal cleavage ions in the same manner as for the b- and y-ions. These two additional ion types are integrated in this third step due to the computational costs that would arise if used in the first step, when scoring all possible sequences.

Confidence Scoring

Lastly, the algorithm converts the refined score into a confidence value. The scores of the best sequence candidates from the previous step are transformed into raw confidence values S_{raw} by following the formula $S_{raw} = \exp(c \times S_{refined})$ with $S_{refined}$ being the refined score and c being a factor approximated from the spectrum. Further normalization is done by mapping the sum of raw confidence values to 1. Confidence for individual amino acids in a sequence is derived from overlapping segments in the sequences with the highest confidence.

3.2.2 Application of the Algorithm

PEAKS can be executed via the software PEAKS Studio 11, for which a licence is needed. It is however possible to request a 15-day trial version, which was used to conduct the de novo predictions in this thesis. [47] When creating a new project in PEAKS Studio, a couple of settings need to be made, to accommodate the spectrum data. These parameters include the enzyme used for proteolytic digestion, the fragmentation method, the instrument type, possible PTMs, and precursor and fragment mass tolerance. PEAKS can work directly with raw files.

3.3 Novor

Described in [26], Novor combines a new scoring function, a dynamic programming approach similar to MSNovo ([30]) and heuristic elements. The novel aspect of scoring candidates incorporates the machine learning technique of decision trees, thus being able to include various ion-types into the decision. Additionally, Novor allows for sequencing of more than 300 MS/MS spectra per second on a conventional laptop computer, while outperforming other state-of-the-art algorithms on four test sets.

3.3.1 Key Concepts

The information in the subsequent paragraphs is a summary extracted from the work of Ma and co-workers [26]. Novor adapted a dynamic programming method used in MSNovo, while also introducing two new scores to assess and fine-tune candidates.

Fragmentation Score

The fragmentation score uses a decision tree to compute a confidence value which describes if a given prefix mass belongs to a fragmentation site. A candidate peptide sequence of length n has $n-1$ fragmentation sites. For each peak of the experimental spectrum that matches one of those sites, 72 fragment ion features and four spectrum features are used. Fragment ion features consist of eight features (e.g., variety of ranks and intensities) for nine different ion types. Spectrum features include the precursor charge as well as the prefix, suffix and peptide mass. By asking questions (i.e., looking at the features) a decision tree advances through its levels until a leave node is reached. The value stored in the leave corresponds to the confidence of the examined fragmentation site. This step is repeated for every fragmentation site. The parameters needed for the decision tree are learned by examining a training data set.

Once all fragmentation sites are scored, the score for the candidate is calculated. The confidence values are summed as described in Equation 3.3. p_i corresponds to the confidence of fragmentation site i. Subtracting 0.1, a factor found empirically, discourages the algorithm from using small amino acids to boost the score.

$$S_F = \sum_{i=1}^{n-1}(p_i - 0.1) \tag{3.3}$$

Residue Score

Novor calculates the probability of a residue in a candidate peptide being correct and uses this score to refine the best performing candidate according to the fragmentation score. Assessing whether a specific amino acid is accurate, this score employs a decision tree. By looking at the left and right neighbour of a residue, a total of 169 features is compiled. Compilation of the residue score can be seen in Equation 3.4. a_i denotes the amino acid at position i in the candidate, $p(a_i)$ equates to its correctness computed with the decision tree and $m(a_i)$ symbolizes the mass the of the amino acid.

$$S_R = \frac{\sum_{i=1}^{n} p(a_i) \times m(a_i)}{\sum_{i=1}^{n} m(a_i)} \tag{3.4}$$

Computation of Peptide Sequence

Pre-computation of the fragmentation score for every possible prefix-mass and multiple runs of a dynamic programming algorithm ensure finding the peptide sequences of good quality. In order to calculate the fragmentation score, the algorithm only needs a specific prefix mass, therefore enabling Novor to compute the score for every possible prefix mass ahead of time. A dynamic programming approach is then used to find the peptide sequence that maximises the score. It is common for some fragment ions to be misinterpreted. Thus, the algorithm is rerun three times with those fragment ions being artificially labeled.

Further refinement of the best scoring sequence is done by dividing it into segments and refining these segments by performing local search. All peptide sequences are scored using the residue score, the best performing one is used for additional refinement. The original sequence is split into segments of a maximum size of 100 residues. Top scoring residues constitute the boundaries of those segments. Local search is carried out for each segment. In each segment, multiple residues are replaced randomly. For the resulting sequences, the residue score is utilized to assess possible improvement. This step is repeated three times.

3.3.2 Application of the Algorithm

The Novor algorithm is included in the software SearchGUI, an application that combines multiple tools used in the field of proteomics. SearchGUI is built in Java, thus a recent Java Runtime Environment (JRE) needs to be installed. [48]

Before any de novo predictions can be made, the software requires that search settings are defined. Parameters like variable and fixed PTM, the type of enzyme, precursor and fragment mass tolerance have to be defined according to the data described in 4.1. In addition to the input file and output directory, a database file has to be provided. This file is used to identify peptides and proteins. However, in this case, an empty file can be used, since no matching to proteins needs to be performed. In addition to the search setting, algorithm specific parameters also have to be set. Novor allows the user to choose the fragmentation method and the mass analyser.

3.4 DirecTag

DirecTag, as described in [31], facilitates the power of statistical scoring. The MS/MS spectrum is interpreted as a graph, with peaks being nodes and amino acids being edges. This approach looks at sequence tags, a set of N nodes connected by consecutive $N-1$ edges. In the paper, the authors focus on a set of four nodes/three edges, providing a good middle ground in terms of computation time and accuracy.

3.4.1 Key Concepts

The following details are a summary compiled from the work of Tabb and co-workers [31]. The algorithm combines three different sub-scores to evaluate every sequence tag. After filtering for the top K peaks in the spectrum, DirecTag looks for peaks that have a distance equal to the mass of one of the 20 amino acids. This information is used to construct a graph. For every sequence tag consisting of N nodes and $N-1$ edges, three sub-scores are calculated. Each sub-score is transformed into a p-value that corresponds to the probability of a better score being achieved by a selection of random peaks. The computation of the scores as well as the p-value transformation are described below. All three p-values are combined into a single test statistic by applying Fisher's Method for combing p-values. Combination is done by applying Equation 3.5 with m being the number of p-values to combine. The combined Fischer's test FCT follows a χ^2 distribution with $2m$ degrees of freedom.[32]

$$FCT = -2\sum_{i=1}^{m} \ln(p_i) \tag{3.5}$$

Intensity Sub-Score

Tags that explain strong peaks tend to be more correct than tags with moderate intensities. However, this sub-score does not make use of the absolute intensity but rather of the intensity rank of a peak. The highest peak in the spectrum occupies rank one, the second-highest rank two, and so forth. All ranks of a sequence tag are summed up, the resulting rank sum is converted into a p-value.

Conversion into a p-value is based on the Mann-Whitney-U test. In its core, this statistical test analyses if the four peaks of the sequence tag originate from an equal distribution as all other peaks. At startup, the probability of every observable rank sum when choosing T nodes from a spectrum with N nodes is computed. For the rank sum of a specific tag, DirecTag is able to look up the probability of an equal or lower rank sum.

Fidelity Sub-Score

The peaks (nodes) in valid sequence tags are separated by the mass of one amino acid. Scoring the quality of such a valid tag facilitates a Summed Squared Error (SSE). Computing the SSE for a tag is done as follows: starting at the second peak (node) all previous amino acid masses are subtracted. For a perfect sequence tag, this should result in the mass of the first peak. However, usually four different masses are retrieved (including the first peak). Each of these four masses is subtracted by the average of the four masses, the differences are squared and summed up, resulting in the SSE.

DirecTag runs a random simulation during the start of the program, allowing for the calculation of the probability that an equal or lower SSE is achieved by chance. This probability is used as the p-value.

Complementary Sub-Score

Peaks of complementary ions are to be considered more reliable than other peaks, thus sequence tags that match one or more complementary ions receive a better score. Complementary ions are found by DirecTag during a pre-processing step, and pointers are stored accordingly. Some tags will exhibit the same amount of matches to complementary ions, thus an SSE is calculated to further distinguish between them. Summation of the masses of a complementary pair estimates the precursor mass. The difference between the calculated and actual precursor mass is squared. All errors for a sequence tag are summed. Sequence tags are primarily sorted by the descending number of complements they match. Secondary sorting uses the SSE in ascending order.

Calculating the p-value for a sequence tag usually consists of two parts that are summed up: (1) the probability of matching more complementary ions by chance than in the current sequence tag and (2) the probability of achieving a lower or equal SSE by chance with the same amount of complementary ions as the current sequence tag. The probability of complementary ions per sequence tag uses the hyper-geometric distribution with N peaks, M of them marked when drawing n times. Achieving a lower or equal SSE by chance is computed via a random simulation during the application start. Summation of these probabilities leads to the p-value. While (2) is used for all sequence tags, (1) is only used for those that did not already match the maximum amount n.

3.4.2 Application of the Algorithm

DirecTag is included in the software SearchGUI. Requirements and usage advice regarding this application can be seen in 3.3.2. The for Novor defined search settings, are used for every algorithm in SearchGUI, and do not to be redefined for DirecTag. However, the algorithm can be adjusted by tuning some parameters, including the tag length and the maximum number of tags to keep. Since the authors of DirecTag claimed that a tag length of 4 seems to provide a good middle ground between accuracy and computation time, this analysis will be using tags of length 4 as well.

Chapter 4

Methods

4.1 Datasets

The data to be analysed was sourced from the project ProteomeTools. The following description of the dataset has been lifted from the work of Zolg and co-workers[33]. The project aims to deliver mass spectrometry data of around 1.4 million synthesized peptides, which corresponds to a human genome coverage of virtually 100%. To achieve this, peptides were selected from ProteomicsDB and the SRMAtlas. Synthesis of the peptides was done in pools. An equal mass distribution of the peptides in these pools is achieved by a computational sorting beforehand.

Synthesized peptides were treated and put through a HPLC system which feeds into an Orbitrap Fusion Lumos mass spectrometer. Each pool was used with three different dissociation types: Higher-energy Collisional Dissociation (HCD), IT and ETD. Data from the mass spectrometry experiment was inspected using the database search software MaxQuant. Based on confidently scored precursors, a list with all included peptides was created for every pool. The results of MaxQuant as well as the inclusion list are part of the dataset.

4.1.1 Selected Subdatasets

For the purpose of this analysis, the raw files of three different pools were selected. All three pools use the same configuration for the mass spectrometer and the analysis with MaxQuant. The spectra were created by using HCD fragmentation on a Fourier transform mass spectrometer (FTMS). For precursor mass tolerance and fragment mass tolerance 0.0045 Da and 0.02 Da were used respectively. MaxQuant analysed the spectra based on these parameters and additionally included the two post-translational modifications, carbamidomethylation of cysteine (fixed) and oxidation of methionine (variable). As a result, a file with all usable spectra is produced, of which a subset was identified by MaxQuant. The proportions of these numbers for each pool can be seen in Table 4.1

4.1.2 Intersection with Predicted Data

After the analysis of the raw data with MaxQuant, the file “msmsScans.txt” is created. This file associates spectra with sequences. Although these sequences are a prediction of MaxQuant, they usually hold very high scores and are thus considered to be accurate. Additionally, the authors of the dataset provide a list of all the peptides that are included in a pool. This list can be found in the file “peptides.txt”. In the light of the analysis, all the predictions made by MaxQuant were checked against this inclusion list, to make sure only correct sequences are used for further analysis. It turns out, all of these sequences are part of the inclusion list, thus no exclusion of rows in “msmsScans.txt” is necessary. Predictions that can not be linked to a MaxQuant identification are checked against the inclusion list.

Table 4.1: Number of spectra and number of MaxQuant identifications per pool. Results of MaxQuant were double-checked with the provided inclusion list. Every spectrum marked as identified was found verbatim in the list. Since all identifications are correct, these numbers will be compared to the results of the de novo algorithms.

Pool	Nr. of Spectra in Dataset	Nr. of Identifications by MaxQuant
Pool 49	50154	33910
Pool 52	49044	31605
Pool 60	49461	31502

4.2 Conversion and Preprocessing

Except for PEAKS, all algorithms require the input as an MGF file, which is briefly described in 2.1.4. Conversion of the raw file can be achieved by SearchGUI and the ThermoRawFileParser [34].

While the regular converted MGF file can be used with Novor and DirecTag, DeepNovo requires further editing of the MGF file. Rearrangements of the spectrum attributes have to be made. DeepNovo demands the following order: TITLE, PEPMASS, CHARGE, SCANS and RTINSECONDS. A different order concludes in uninterpretable predictions. Furthermore, the attribute SEQ has to be added after RTINSECONDS. Although SEQ is only used in training to provide the actual sequence of a spectrum, the documentation still advises adding it to test data. The assigned value, however, has no meaning. For the prediction made in this project, SEQ=PEPTIDE is used.

4.3 Parsing of Algorithm Results

All four de novo algorithms put their results in a tabular format, but produced values as well as spelling of the columns vary. Custom parsers for each algorithm result provide the flexibility to handle each file differently, while maintaining a common output structure suitable for further analysis. Figure 4.1 A shows the structure used for implementing parsers for all algorithms.

4.3.1 Ideal Structure for further Analysis

The nature of all the scoring schemes described in 4.4 is to compare the predicted peptide sequence created by the algorithms to the actual peptide sequence. Furthermore, all four algorithms produce some sort of score for their predictions. It is of interest to include this scores in the analysis. Thus, the algorithm results are reduced to the predicted sequence, the actual sequence, a scan number or index and the algorithm score. If possible, the actual sequence can be retained from the experiment data described in 4.1. The scan number or index allows for correct association of the predicted sequence with a MaxQuant result. All the columns that are not used, contain additional information that will not be used in the scope of this analysis.

4.3.2 DeepNovo and PEAKS

DeepNovo and PEAKS each produce a tabular result files (TSV and CSV respectively) without any additional headers. The Python library Pandas [49] was used to parse the data. The columns “Scan”, “Peptide” and “local confidence (%)” of the PEAKS result and the columns “scan”, “output_seq” and “output_score” of the DeepNovo results are selected for further usage. However, in both cases some processing is needed. The DeepNovo prediction has to be converted from a list of letters, separated by commas, to a single string. The score, calculated by PEAKS, gives a confidence for each position ranging from 0% to 100%. For the sake of the analysis, the average of the position specific score is computed.

In either case, the predicted peptide sequence can contain post-translational modifications (PTMs) that need to be adjusted. DeepNovo appends the suffix “mod” to the amino acid that was modified. “Mmod” would therefore be the oxidation of methionine. PEAKS on the other hand includes the additional mass of the PTM inside of parentheses after the amino acid. In the peptide sequence “LLPSC(+57.02)LH” the amino acid cysteine is modified. Both styles allow for simple parsing.

Since both algorithm results include the scan number of the corresponding spectrum, straightforward merging with the related MaxQuant result is possible. The Python library Pandas [49] is used to join the reduced results and the expected peptides on the column “Scan”, with the outcome being exported as a TSV for later scoring.

4.3.3 DirecTag and Novor

DirecTag and Novor produce tabular result files (TSV format), but both include multiple lines of information about the algorithm parameters before the actual data is described. In the case of Novor, these lines can be skipped via using the Python library Pandas [49]. For DirecTag however, information about the PTMs needs to parsed, therefore these lines cannot be skipped. Modified amino acids in a sequence tag are replaced by a number. The header defines which modification is mapped to which number. The substring “DynamicMods: M 0 15.994915” contained in one of the headers, defines that an oxidized methionine is substituted by the number 0. The third argument can be ignored for the purpose of this analysis.

Furthermore, the actual result data of DirecTag distinguish between two different row types. Lines of type “S” correspond to a certain spectrum, while lines of type “T” correspond to a sequence tag. DirecTag produces multiple sequence tags per spectrum, of which the top scoring N are kept, with N being an adjustable parameter of the algorithm. Therefore, all the sequence tags that follow a line of type “S” are to be seen as the top scoring candidates for this spectrum. This unique structure does not allow for the use of the Pandas [49] library. Instead, all the lines of the file are iterated and a list entry for each tag with the corresponding spectrum ID, DirecTag score and tag sequence is made.

Given that both algorithms include PTMs, the predicted sequences need to be adjusted before further usage. Since Novor includes the modification as a number enclosed in parentheses, the modifications can be excluded by only using the alphabetic symbols of the prediction. Using the parsed mapping from above, every number in a DirecTag sequence tag is replaced by the modified amino acid it represents.

The columns “id”, “peptide” and “score” of the Novor result and the spectrum ID, tag sequence and score of the DirecTag results are further used. Due to both algorithms ignoring the scan number of the MS-data, merging with the corresponding MaxQuant result has to be done by joining on the index. The outcome is saved to a TSV file for later scoring.

4.4 Scoring Methods

For all the described scoring methods below, a Python class was implemented. A concrete scorer extends an abstract base class. Figure 4.1 B shows what scorers exist and how they are structured.

4.4.1 (Normalized) Alignment Score

Comparing two or more sequences is a very common task in the field of bioinformatics. Applying a so-called alignment method is a crucial step in finding common genes in different species and predicting the structure of a protein. [35] Alignment methods try to fit two strings together in a way that a score is optimal. To achieve this optimal score, it is sometimes necessary to open or extend gaps that come along with a penalty. Additionally, when comparing protein sequences, substitutions matrices like BLOSUM or PAM are used. [36] These matrices hold scores for replacing one amino acid with another while respecting evolutionary and physio-chemical factors. [37] In the case of two sequences, a pair wise alignment can be made in two fashions: global or local. While a global alignment tries to match the entirety of both sequences, a local alignment tries to align a substring of one sequence with a substring of the other sequence. [36]

For the purpose of this analysis, the local and global alignment is constructed using

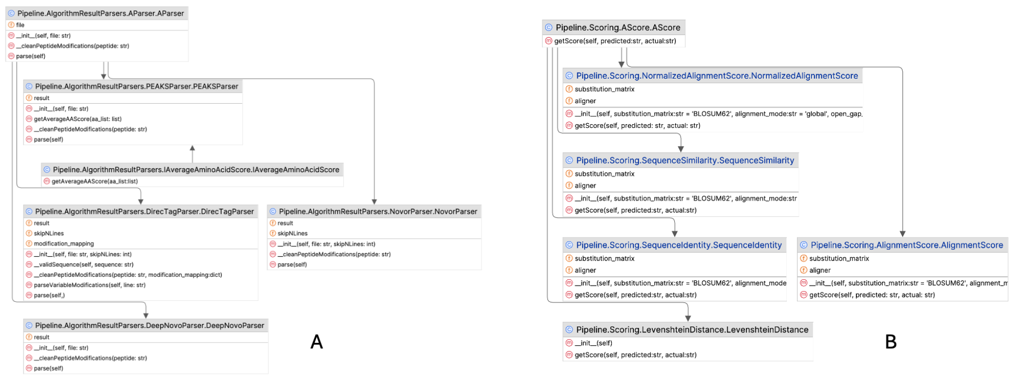
Pipeline.AlgorithmResultParsers.AParser.AParser
file
__init__(self, file: str)
__cleanPeptideModifications(peptide: str)
parse(self)
Pipeline.AlgorithmResultParsers.PEAKSParser.PEAKSParser
result
__init__(self, file: str)
getAverageAAScore(aa_list: list)
__cleanPeptideModifications(peptide: str)
parse(self)
Pipeline.AlgorithmResultParsers.IAverageAminoAcidScore.IAverageAminoAcidScore
getAverageAAScore(aa_list:list)
Pipeline.AlgorithmResultParsers.DirecTagParser.DirecTagParser
result
skipNLines
modification_mapping
__init__(self, file: str, skipNLines: int)
__validSequence(self, sequence: str)
__cleanPeptideModifications(peptide: str, modification_mapping:dict)
parseVariableModifications(self, line: str)
parse(self,)
Pipeline.AlgorithmResultParsers.NovorParser.NovorParser
result
skipNLines
__init__(self, file: str, skipNLines: int)
__cleanPeptideModifications(peptide: str)
parse(self)
Pipeline.AlgorithmResultParsers.DeepNovoParser.DeepNovoParser
result
__init__(self, file: str)
__cleanPeptideModifications(peptide: str)
parse(self)
A
Pipeline.Scoring.AScore.AScore
getScore(self, predicted:str, actual:str)
Pipeline.Scoring.NormalizedAlignmentScore.NormalizedAlignmentScore
substitution_matrix
aligner
__init__(self, substitution_matrix:str = 'BLOSUM62', alignment_mode:str = 'global', open_gap.
getScore(self, predicted: str, actual: str)
Pipeline.Scoring.SequenceSimilarity.SequenceSimilarity
substitution_matrix
aligner
__init__(self, substitution_matrix:str = 'BLOSUM62', alignment_mode:str
getScore(self, predicted:str, actual:str)
Pipeline.Scoring.SequenceIdentity.SequenceIdentity
substitution_matrix
aligner
__init__(self, substitution_matrix:str = 'BLOSUM62', alignment_mode
getScore(self, predicted:str, actual:str)
Pipeline.Scoring.AlignmentScore.AlignmentScore
substitution_matrix
aligner
__init__(self, substitution_matrix:str = 'BLOSUM62', alignment_m
getScore(self, predicted: str, actual: str)
Pipeline.Scoring.LevenshteinDistance.LevenshteinDistance
__init__(self)
getScore(self, predicted:str, actual:str)
B

Figure 4.1: Class hierarchy for parsers and scoring methods. **(A)** Concrete parsers for every algorithm extend the abstract base class "AScorer". The purpose of every concrete parser is to handle the different results format produced by the algorithms. In addition to "AScorer", the interface "IAverageAminoAcidScore" is implemented in the PEAKS parser. Since peaks provides scores for every position in the sequence, this has to be handled. The other three algorithms provide an overall score in their results. **(B)** Every scoring metric defined in this work is implemented as a concrete class. All scorers extend the abstract base class "AScorer" which defines a method to calculate the score between two sequences. Scores that rely on alignments have default values set in their constructor for penalty parameters and substitution matrix.

the predicted and actual sequence. The implementation for both is provided by the Python library Biopython. If these two sequences can be aligned, the score of the optimal alignment is used. Should no alignment be possible, the value 0.0 is used. Costs for opening and extending gaps have a default value of -2. The default substitution matrix is BLOSUM62. The alignment score of longer sequences tends to be higher, which makes the raw alignment score not really suitable for a comparative analysis. Hence, the global and local alignment scores are normalized by dividing them by the length of the alignment including gaps.

4.4.2 Percent Identity

Building on the result of a pairwise alignment, the sequence identity of two sequences can be calculated. The identity of two sequences is defined as the identical positions in an alignment. Since sequences do not need to be of the same length, it is common to express the identity in a percentage value. The approach used for this analysis utilizes the length of the alignment to calculate the percentage. [38] The formula for the percent identity (PID) can be seen in Equation 4.1. Default values for the gap open and extension penalty are set to −2. The default substitution matrix is BLOSUM62. Due to DirecTag only predicting substrings (tags), identity computation uses local alignment with gap open and extension costs of −10. This prevents the association of the tag all over the actual sequence. In addition to that, the score is divided by the length of the tag, and not the length of the alignment. This ensures that tags with only three matching positions receive a percent identity score of 0.75 at a tag length of four. A perfect match between two sequences will lead to a score of 1.0.

$$PID = \frac{N_i}{|A|} \tag{4.1}$$

where:

N$_i$ = number of identical positions of the alignment
$|A|$ = length of the alignment

4.4.3 Percent Similarity

While the percent identity only accounts for identical positions in an alignment, it is also of interest how similar the sequences are. A prominent example is the use of leucine instead of isoleucine. Both have the same mass and cannot be discriminated in a mass spectrometer. For this purpose, the so-called sequence similarity can be used. Using protein matrices like BLOSUM or PAM, two amino acids can be seen as similar, when their substitution score is positive. [39] To make this measure more comparable, a percent value is used. The formula used to calculate the percent sequence similarity (PSI) can be seen in Equation 4.2. Default values of gap open and extend penalty, used in the alignment, are set to −2. By default, the BLOSUM62 matrix is used. DirecTag is treated the same as in the percent identity score calculation. A perfect match between two sequences will lead to a score of 1.0.

$$PSI = \frac{N_p}{|A|} \tag{4.2}$$

where:

N$_p$ = number of positive substitution scores of the alignment
$|A|$ = length of the alignment

4.4.4 Levenshtein Distance

The Levenshtein distance is an approach based on the Hamming distance. Hamming distance measures how many substitutions are necessary to turn string A into string B. Levenshtein expanded on that concept, by also allowing insertions and deletions. In the context of Biology, the Levenshtein distance gives insight to how closely two sequences are related. The calculation is done via a dynamic programming algorithm. [40] For this analysis, the Python package Levenshtein [50] was used.

4.5 Analysis

Scored results are analysed using Jupyter Notebooks [41] and the programming language Python [42]. Data is loaded as a data-frame using the Pandas package [49]. Bar-charts and box-plots are created via the package Matplotlib [43], scatter plots are generated with the Seaborn package [44]. Correlation in the form of R^2 is calculated using the package SciPy [45].

Chapter 5

Results

The entire source code used for the analysis can be found in the GitHub repository [46] associated with this project.

5.1 Search Parameters and Pipeline

5.1.1 Search Parameters

As described in the application section of every algorithm (see 3.1.2, 3.2.2, 3.3.2 and 3.4.2), multiple parameters can be adjusted. To accommodate the method with which the data was collected, these parameters are set to the corresponding values described in 4.1.1. If not mentioned otherwise, the default parameters were used. This includes, but is not restricted to, the maximum m/z value for a spectrum in DeepNovo and the application of preprocessing steps in PEAKS.

5.1.2 Description of Pipeline

The algorithms, with the parameters outlined above, were applied to three different pools of the dataset described in 4.1. In the cases of Novor, DirecTag and DeepNovo, the provided raw files had to be converted to the MGF format. Additionally, DeepNovo required further preprocessing steps (see 4.2). The resulting algorithm outputs were parsed, merged with the actual sequences and stored. Most predictions could be merged with a MaxQuant result. Based on these files, the scores described in 4.4 are computed. Those who were not identified by MaxQuant are checked against the inclusion list by finding sequences with identity or similarity scores of 1.0. These metrics give insight into how similar the predicted and the actual sequence actually are.

5.2 DeepNovo

5.2.1 Number of Predictions

DeepNovo was able to predict around 80% of the spectra throughout the three pools. Only 60% of all spectra could be scored using the metrics. DeepNovo had some difficulties handling Pool 60. While both Pool 49 and 52 exhibit predictions for 85% of all spectra, only 69% of Pool 60's spectra are covered by the algorithm's predictions. The influence of this fact can be seen throughout the analysis of DeepNovo. A possible explanation for this is discussed in 6.

Table 5.1: Number of predictions made by DeepNovo and number of predictions that could be scored. DeepNovo was able to make predictions for around 80% of all spectra for every pool. The algorithm excludes spectra based on their properties. However, only around 60% of all spectra could be linked to a sequence and thus could be score.

Pool	Nr. of Spectra in Dataset	Nr. of Predictions made by Algorithm	Nr. of Spectra that could be scored
Pool 49	50154	43014	31560
Pool 52	49044	42127	30390
Pool 60	49461	34608	26579

5.2.2 Identity and Similarity

Based on the identity and similarity of scored sequences, the median scores fluctuate across all pools. The median percent identity scores are at 0.66, 0.70 and 0.75 for the pools 49, 52 and 60 respectively. In the distribution of the percent similarity, the median scores increase to 0.81, 0.84 and 1.0. Considering all three pools, DeepNovo was able to completely identify about 8% of all spectra using the identity metric. Around 7% are attributed to the MaxQuant identification, the remaining 1% are found by checking the inclusion list. Employing the similarity score, however, lead to a correct prediction of approximately 28% of all spectra. About 24% were identified by using the MaxQuant results, 4% were found upon reviewing the inclusion list. On average, this equates to an increase by a factor of 3.5. Performances of both scores for every pool can be seen in Figure 5.1.

5.2.3 Levenshtein Distance

The median Levenshtein distances measured in Pools 49, 52 and 60 are 4, 3 and 2. Furthermore, 50% of all predictions for Pool 60 show a score in the range of 1 to 5, whereas Pools 49 and 52 have an interquartile range (IQR) of 2 to 8 and 2 to 7 respectively. Box-plots of every pool's distribution can be seen in Figure 5.2. Pool 60 presents the best quality of predictions in terms of the Levenshtein distance, since fewer changes are needed to reach the actual sequence compared to the other two. However, this type of behavior is probably caused by the lower number of predictions made by DeepNovo for Pool 60 (see 5.1).

5.2.4 Algorithm Score

DeepNovo produces a score to assess the prediction it made. Over all three pools, this score ranged from -3.28 to 0.00, with 0.00 being the best score. Interquartile ranges go

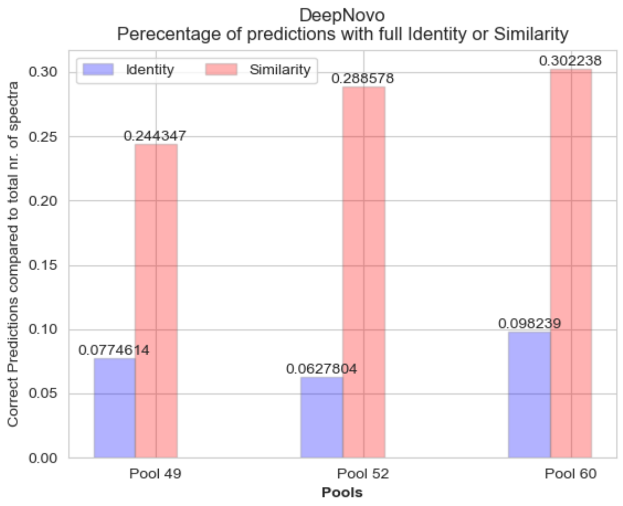

Figure 5.1: Correct predictions in terms of identity and similarity, in relation to the total number of spectra per pool.

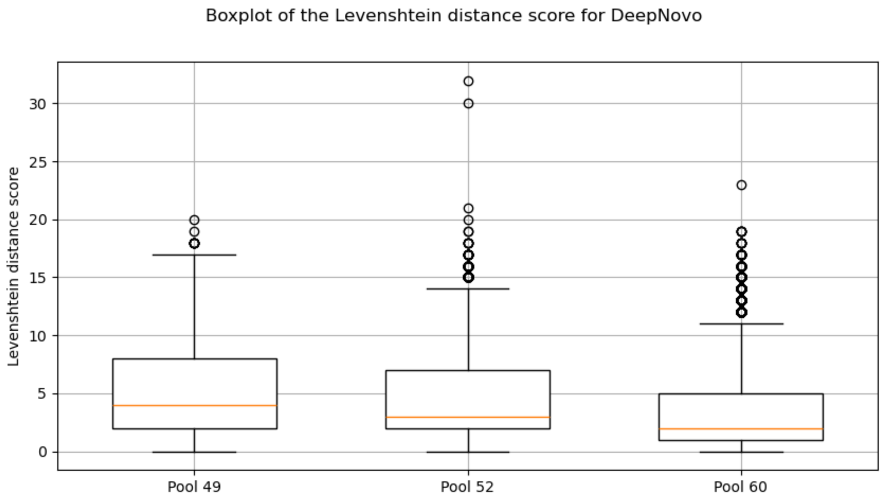

Figure 5.2: Levenshtein distance of DeepNovo results for each pool.

from -0.41 to -0.03, the median is equal to -0.19. Figure 5.3 shows how the DeepNovo score and identity or similarity metric correlate. A higher score, results in a more accurate prediction. However, the score is only somewhat reliable since the R^2 values are 0.49 and 0.55 respectively.

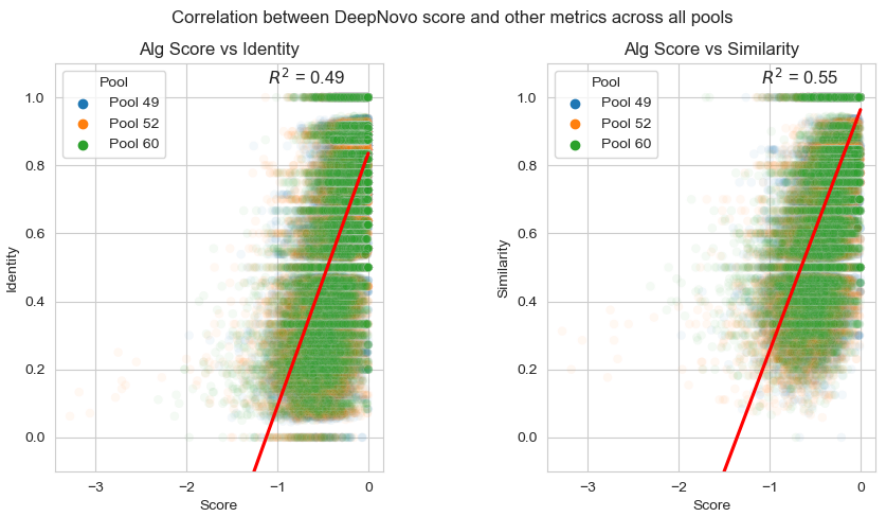

Figure 5.3: Scatter plots of the score computed by DeepNovo compared to the metrics identity and similarity. The linear regression (red line) and R^2 are calculated with the Python package SciPy. R^2 values of 0.49 and 0.55 for identity and similarity, respectively, indicate that these two metrics roughly correlate with the predicted score from DeepNovo.

5.3 Peaks

5.3.1 Number of Predictions

On average, PEAKS was able to predict approximately 80% spectra of the combined three pools. The analysis is able to score around 65% of all spectra. Absolute values for each pool are depicted in Table 5.2. Similar to DeepNovo, the percentage of predictions across the three pools varies a bit. PEAKS is able to provide peptides for 72% of the spectra in Pool 49. Pool 52 and 60 however, are covered by 82% and 86% of the predictions.

Table 5.2: Number of predictions made by PEAKS and number of predictions that could be scored. PEAKS was able to make predictions for around 80% of all spectra for every pool. However, only around 65% of all spectra could be linked to a sequence and thus could be score.

Pool	Nr. of Spectra in Dataset	Nr. of Predictions made by Algorithm	Nr. of Spectra that could be scored
Pool 49	50154	36399	30787
Pool 52	49044	42289	33444
Pool 60	49461	41049	33064

5.3.2 Identity and Similarity

The median scores of percent identity remain stable across all three pools. Pool 49 is at a median of 0.875, while Pools 52 and 60 have a median of 0.857. For the percent similarity, PEAKS was able to achieve a median score of 1.0 across the board. This is further supported by the number of peptides that could be identified. Figure 5.4 shows how many predictions per pool can be considered correct using the identity and similarity metrics. On average, the identity score allows for 20% of all spectra to be recognized as correctly predicted. Around 18% were identified based on complete identity with a MaxQuant identification. The remaining 2% were found via the inclusion list. Using the similarity score expands that to around 38%. About 33% shared full similarity with a MaxQuant entry, 5% could be discovered due to the inclusion list. This increase corresponds to an increase by a factor of 1.9. Both, the median values and the proportional correct identifications, show that PEAKS seems to be a stable algorithm.

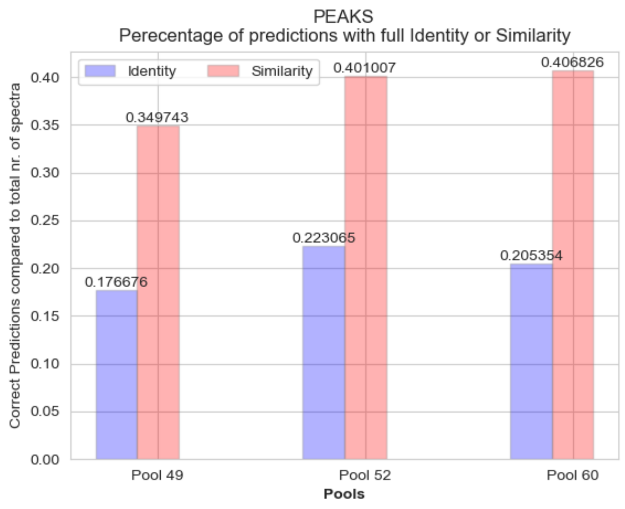

Figure 5.4: Proportion of sequences that were accurately predicted according to the metrics identity and similarity.

5.3.3 Levenshtein Distance

As described in Figure 5.5, the Levenshtein distance for the results of PEAKS, are almost identical across all three pools. Interquartile ranges go from 0 to 3 in every pool. Median scores for Pool 49 and 52 are at a distance of 2, while the median distance of Pool 60 is 1. All three pools further validate the argument, that PEAKS is a very stable algorithm. Due to the very low median of the Levenshtein distance, the quality of the predictions can be considered as good.

5.3.4 Algorithm Score

PEAKS computes a confidence value ranging from 0 to 100 for every position in the predicted sequence. The average positional confidence is calculated, scaled to the range

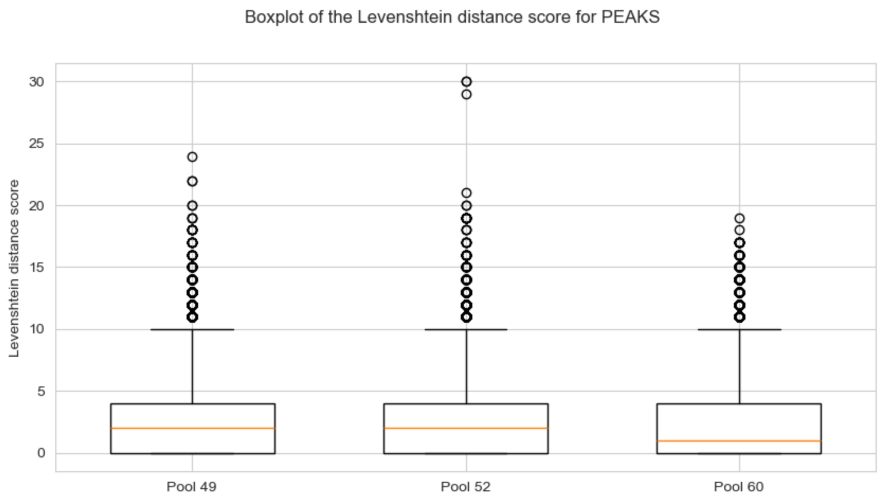

Figure 5.5: Levenshtein distance scores for each pool based on the results of PEAKS. The nearly similar distributions support the fact that the algorithm operates persistently and predictions showcase a very good quality.

of 0 to 1 and used for additional analysis. The lowest value observed over all three pools was at 0.49875, the highest at 1.0. Interquartile distance ranges from 0.72 to 0.96, the median score was at 0.88. The relationships between the average position confidence and the identity and similarity score, are depicted in Figure 5.6. R^2 values of 0.38 and 0.41 show that a mediocre correlation is present.

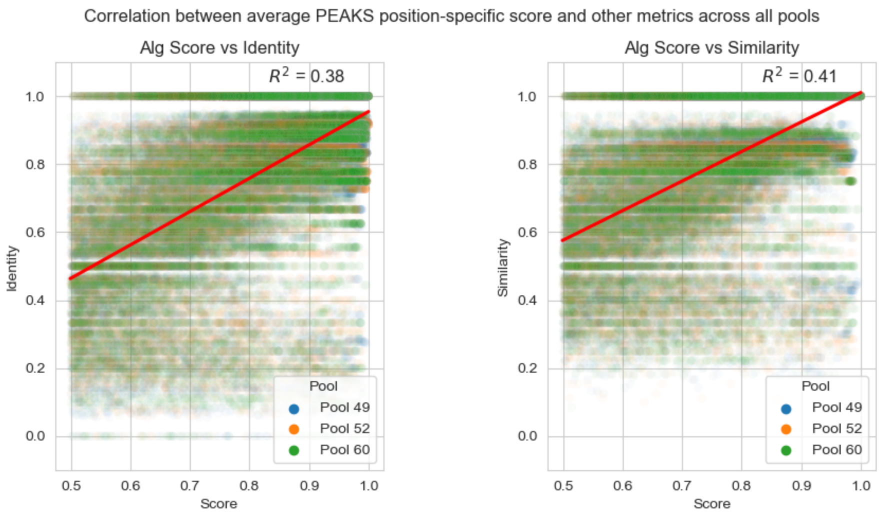

Figure 5.6: Scatter plots of the score computed by PEAKS compared to the metrics identity and similarity. The linear regression (red line) and R^2 are calculated with the Python package SciPy. R^2 values of 0.38 and 0.41 for identity and similarity, respectively, indicate that these two metrics somehow correlate with the predicted score from PEAKS.

5.4 Novor

5.4.1 Number of Predictions

Considering all three pools, Novor was able to predict 99% of the spectra, making it the algorithm with the highest average prediction rate. This is also reflected in the average predictions that can be identified, which lies at 70%. The absolute values of the Novor predictions per pool is illustrated in Table 5.3.

Table 5.3: Number of predictions made by Novor and number of predictions that could be scored. Novor was able to make predictions for around 99% of all spectra in all three pools. Around 70% of all spectra could be linked to a sequence and thus could be score.

Pool	Nr. of Spectra in Dataset	Nr. of Predictions made by Algorithm	Nr. of Spectra that could be scored
Pool 49	50154	50115	36071
Pool 52	49044	49026	34928
Pool 60	49461	49452	34051

5.4.2 Identity and Similarity

The median percent identity and percent similarity scores for the Novor predictions remain rather constant across all pools. Median identity is at 0.833 for Pools 49, 52 and 0.84 for Pool 60 respectively. The median similarity score for Pool 52 and 60 is at 1.0, while the one for Pool 49 is at 0.93. Novor was able to correctly predict around 19% of the all spectra according to the identity score. Looking at the results of the similarity score, however, shows that around 38% of its predictions can be seen as fully accurate. This corresponds to an average increase by a factor of 2. The proportions of accurate predictions made for every pool are shown in Figure 5.7. The total number of predictions consists of two parts. Averaged across the three pools, predictions that can be associated with the corresponding MaxQuant identification make up around 17% and 32% of the predictions with full identity and similarity score. The remaining 2% and 5% are identified by linking the predictions that do not have a MaxQuant identification to peptides in the inclusion list.

5.4.3 Levenshtein Distance

The distribution of the Levenshtein distance scores across all three pools remains similar. Median score is at a distance of 2 in every pool. Pool 60 has the narrowest interquartile range, spanning from 0 to 5. Pool 49 and 52 range from 1 to 6 and 0 to 6 accordingly. With this in mind, it can be said, that the median score and similar IQRs speak for a good quality of the predictions and stability of the algorithm.

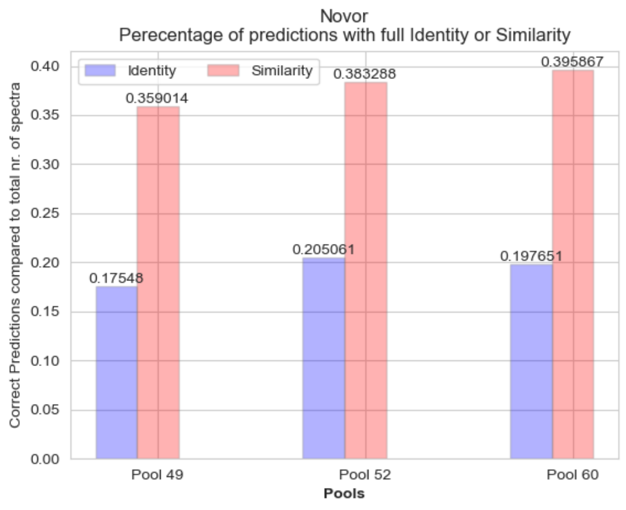

Figure 5.7: Percentage of sequences per pool that were fully predicted according to the metrics identity and similarity.

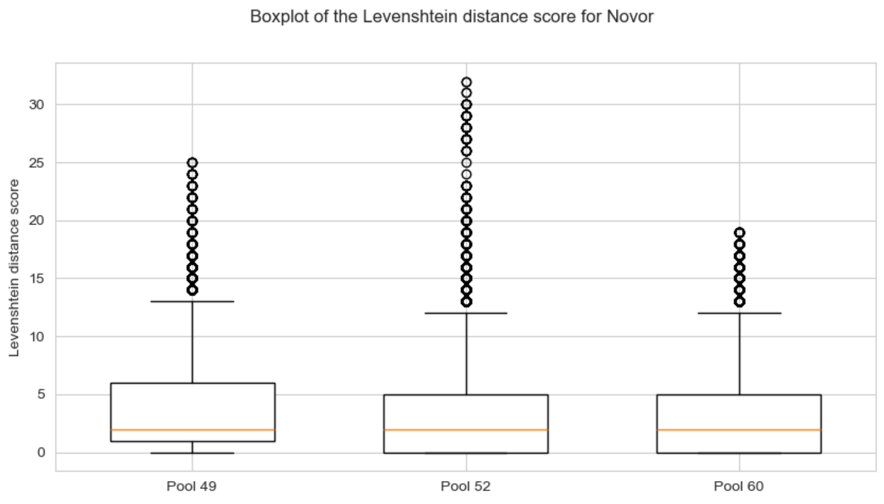

Figure 5.8: Levenshtein distance scores of the Novor results for each pool.

5.4.4 Algorithm Score

For each amino acid sequence predicted, Novor computes a score resembling the quality of the sequence. Looking at all three pools, the lowest occurring values is 1.5, the highest is at 99.4. Half of the scores span from 50 to 89.8 (IQR). The median score is at 74.7. Figure 5.9 shows how the algorithm score relates to the identity and similarity metrics. With R^2 values of 0.63 and 0.66 for identity and similarity, a higher score usually leads to a more accurate prediction of the actual peptide.

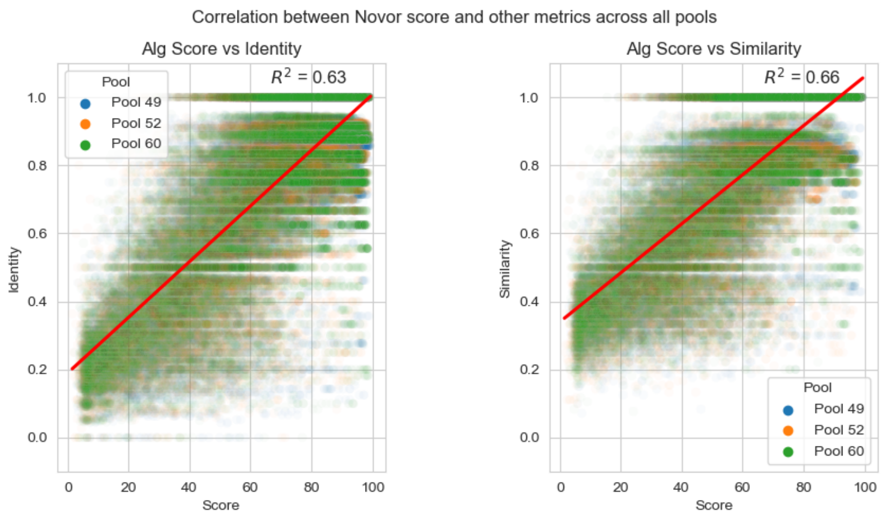

Figure 5.9: Scatter plots of the score computed by Novor compared to the metrics identity and similarity. The linear regression (red line) and R^2 are calculated with the Python package SciPy. R^2 values of 0.63 and 0.66 for identity and similarity, respectively, provide the best correlation of all four algorithms.

5.5 DirecTag

5.5.1 Number of Predictions

Between all three pools, the DirecTag algorithm was able to predict tags for 75% of all spectra, on average. Virtually all of those predictions were eligible for further scoring and analysis. Since DirecTag usually predicts multiple tags for one spectrum, only the number of unique spectra was considered for this calculation. Absolute numbers of spectra and predictions are depicted in Table 5.4.

Table 5.4: Proportion of sequences that were accurately predicted according to the metrics identity and similarity. Both scores show that the algorithm performs steady across every pool.

Pool	Nr. of Spectra in Dataset	Nr. of Predictions made by Algorithm	Nr. of Spectra that could be scored
Pool 49	50154	38291	38247
Pool 52	49044	37284	37220
Pool 60	49461	36174	36110

5.5.2 Identity and Similarity

While all other algorithms predict only one sequence per spectrum, DirecTag produces up to 10 tags for one spectrum. If not handled accordingly, this imbalance in the sheer amount of predictions would skew the analysis results. The percent identity and percent similarity of one spectrum is calculated by averaging the identity and similarity score of all tags associated with one spectrum.

Median identity of the three pools is at 0.625, 0.575 and 0.6 for the Pools 49, 52 and 60 with interquartile ranges typically spanning from around 0.45 to 0.75. Median similarity is located at 0.85 for every pool. IQR of the similarity score starts at 0.7 on average and goes up to 1.0. Since most of the identity values are below a score of 0.75, using percent identity for identification, results in only 4% of accurate predictions. 1.4% come from association with MaxQuant identifications, the remaining 2.6% were found by mapping predictions without MaxQuant identification to the inclusion list. However, introducing the similarity metric, caused a substantial increase in the accurate predictions. On average, 24% of all spectra could be mapped to a prediction. Around 3.6% were identified by looking at MaxQuant results, the other 20.4% were identified by mapping the tags to the inclusion list. This improvement corresponds to an average increase by a factor of 6. The performance of the two metrics for every pool can be seen in 5.10. A penalty to open and extend a gap of -10 is used for identity and similarity. This is to keep the local alignment, which is used in their computation, from opening too many gaps and associating the tag all over the actual peptide. The huge increase in the inclusion list association, compared to the other algorithms, is likely attributed to the use of substrings. Especially since the tags have a length of four amino acids, they will likely be similar to substrings of many peptides.

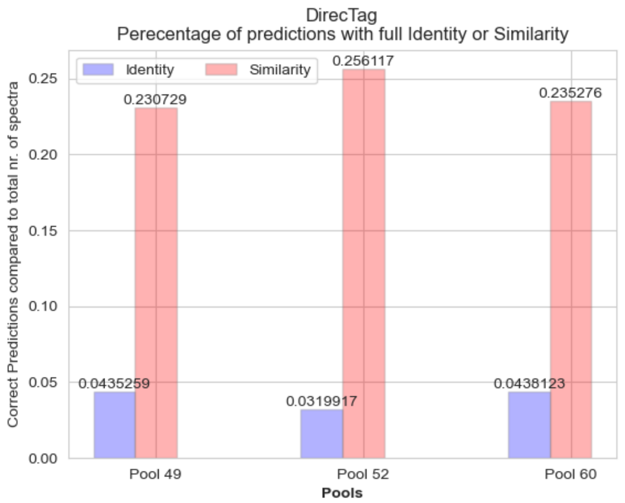

Figure 5.10: Number of accurately predicted peptides, compared to the total number of spectra, shown for each pool. The immense rise in correct predictions, when using the similarity metric, is potentially linked to the fact that the peptides of the inclusion list are queried for substrings similar to tags not identified by MaxQuant. Since tags have a length of four amino acids, it is very likely that a lot of peptides have sections that will lead to full similarity with one tag.

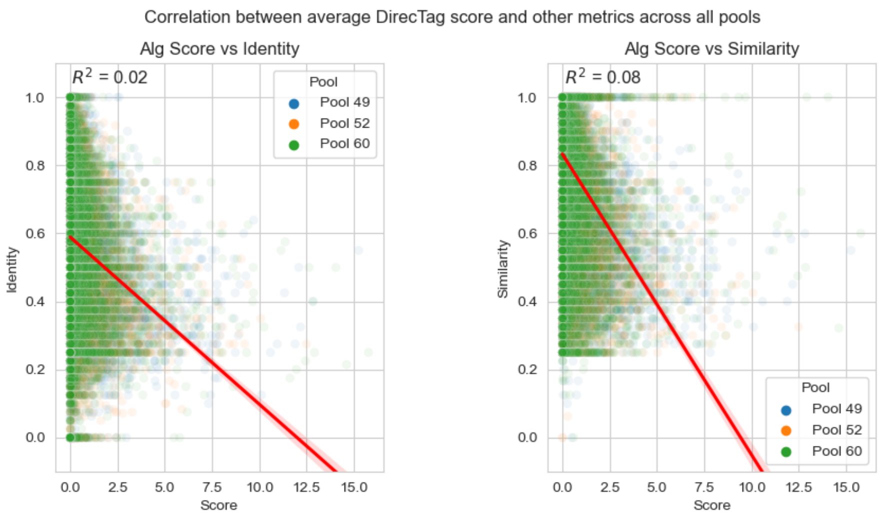

Figure 5.11: Scatter plots of the average combined p-value computed by DirecTag compared to the metrics identity and similarity. The linear regression (red line) and R^2 are calculated with the Python package SciPy. R^2 values for identity (0.05) and similarity (0.08) show that the p-value does not correlate to these two metrics.

5.5.3 Levenshtein Distance

Computation of the Levenshtein distance was skipped for DirecTag's sequence tags. Since only substrings of length four are produced, a lot of edits are necessary to turn the sequence tag into the actual sequence.

5.5.4 Algorithm Score

DirecTag combines three individual sub-scores in the form of p-values into a single p-value. The smaller the p-value, the unlikelier it gets to produce the corresponding tag by chance. Since DirecTag predicts multiple tags, the average of the p-values for one spectrum is used here. Across all three pools, the lowest and highest average p-values that appeared are located at 0.0 and 15.774. The median of p-values is at 0.004769, interquartile ranges extend from 0.000091 to 0.058686. Plotting the average p-value versus the identity and similarity score in Figure 5.11 shows R^2 values of 0.02 and 0.08. Subsequently, it is to say, that there is no correlation between the average p-value and the analysed metrics.

Chapter 6

Discussion

6.1 Comparison of Results

Based on the results presented above, the four algorithms can be compared against each other on different aspects.

Number of Predictions made

None of the programs made predictions for every single one of the spectra. However, Novor covers each pool to more than 99%. DeepNovo and PEAKS tied for second most predictions. While both of them each dominate one of the three pools, the number of their predictions in Pool 52 is virtually equal. Although DirecTag has the third most predictions in Pool 49 and 60, on average it comes in last. Due to the fact, that DirecTag can produce multiple tags for a single spectrum, only the unique number of spectra for which at least one tag exists was used in this case. Figure 6.1 shows the absolute number of predictions for every pool, including the size of the pool itself.

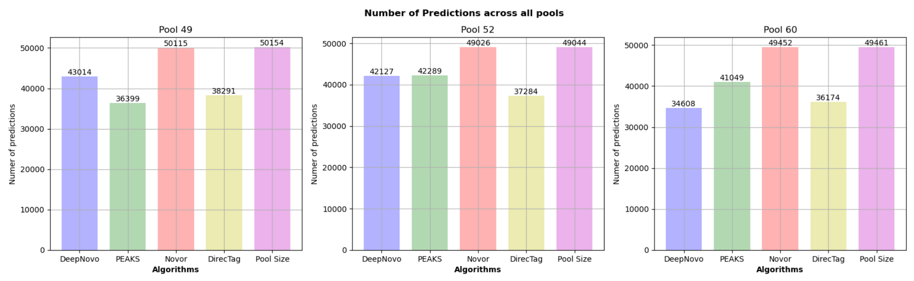

Figure 6.1: Number of predictions made by the four algorithms and the size of the pool (purple). Novor clearly made the most predictions, by covering more than 99% of the pool's spectra. PEAKS and DeepNovo each dominate one pool, but are virtually equal in the third. On average, DirecTag made the least predictions.

Number of Identified Spectra

The authors of the data set provide a list of spectra. A substantial number of them were identified by MaxQuant and thus have sequences associated to it. Just like MaxQuant, all four algorithms produced predictions for only a subset of the available spectra. The predicted sequences of the four algorithms can be associated with an entry in the MaxQuant result list. For the prediction to be considered accurate, the percent similarity score of the prediction and the MaxQuant identification has to be 1.0. However, predictions that do not have a MaxQuant identification can still be accurate. For this reason, these sequences are checked against the inclusion list. Consequently, the number of predictions with 100% similarity in Figure 6.2 includes identifications from both sources. Additionally, the pool size and the number of spectra identified by MaxQuant are shown. Unfortunately, no de novo identification algorithm could beat the performance of database searches in the form of MaxQuant. Despite that, a hierarchy of the de novo algorithms in terms of their identification performance can be compiled. On average, PEAKS was able to identify slightly more spectra than Novor. DeepNovo comes in third and DirecTag comes in last. For DirecTag, only the spectra with an average tag similarity score of 1.0 are counted.

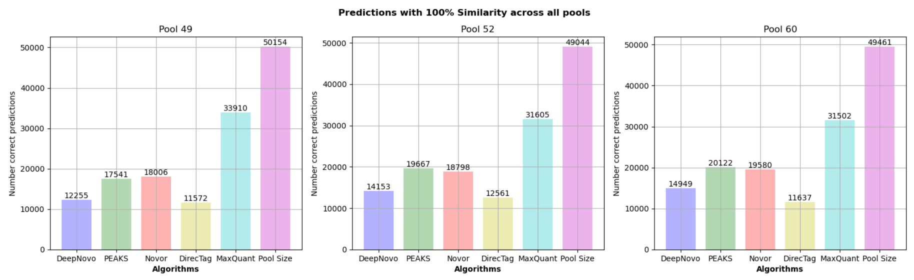

Figure 6.2: Number of predictions with a similarity score of 1.0 compared to the number of spectra identified by MaxQuant (cyan) and the actual pool size (purple). Even the two best de novo approaches, PEAKS and Novor with essentially the same number of accurate predictions, made considerably fewer accurate prediction compared to MaxQuant.

Quality of Predictions

In the process of aligning two sequences, a score is calculated. Longer sequences that match each other will result in a higher score. To counteract this behaviour and in order to make scores comparable, they are normalized by dividing through the length of the alignment. Since DirecTag can only predict sequence tags i.e. substrings of a sequence, only local alignments should be used. The normalized score of the optimal local alignment can be used to assess the quality of the predictions made by the algorithms. Figure 6.3 shows the distribution of these scores. The predictions made by the algorithms PEAKS and Novor show a left skewed distribution. This means that these two algorithms predicted sequences that are very much in line with the actual sequences identified by MaxQuant or found in the inclusion list. DeepNovo's predictions result in bimodal distributions, with a weak peak centering around the score 1 and a strong peak centering around the score 5. While DeepNovo made a lot of predictions that are consistent with the actual sequences, a substantial amount of predictions was also of low quality. DirecTag exhibits a distribution with only one peak, centering around the score 3.5. Although DirecTag performs worse than the other algorithms, the mono-modal distributions appear to be more stable than the bimodal distribution of DeepNovo.

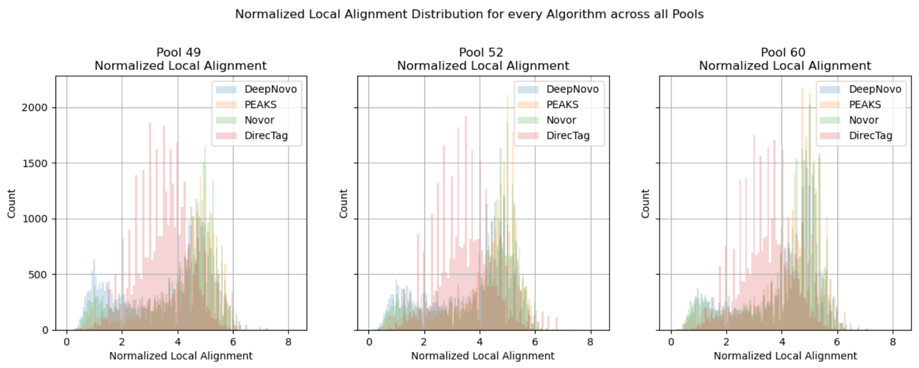

Figure 6.3: Distribution of the normalized local alignment scores for every algorithm in each pool. DirecTag stands out with most of the predicted tags centering around the score of 3.5. Novor and PEAKS performed very well, since most of the scores are focused around a score of 5. DeepNovo exhibited a similar behaviour, although also showing an increased amount of scores centred around 1.

6.2 Possible Explanation for Differences between Algorithms

6.2.1 Distribution of Pools

According to the creators of the dataset, all pools should exhibit a similar distribution of the peptide mass. However, the analysis shows that Pool 60 holds predictions with better quality. By examining the mass distribution of the inclusion list of each pool, a possible explanation for this behaviour can be provided. Figure 6.4 shows histograms and box-plots for all three experiment pools. While the distributions of Pool 49 and 52 look related, Pool 60 and its very strong bimodal shape stand out. When looking at the median values and IQRs, the differences between Pools 49 and 52 also become more apparent.

6.2.2 Difficulties regarding DeepNovo

There are two reasons to which DeepNovo has performed worse than algorithms like PEAKS and Novor. For one, the pretrained model, which is linked in the project's GitHub repository [28], was used. This model was trained on data from yeast, however, the exact species is not mentioned.
On the other hand, the algorithm excludes spectra from the analysis based on a computation that is not in line with the instructions provided in the above repository. Due to the fact, that DeepNovo can be used to train a custom network, the input MGF file is required to have the attribute SEQ, which holds the actual peptide sequence. According to the repository, for de novo sequencing, an arbitrary sequence can be provided for the

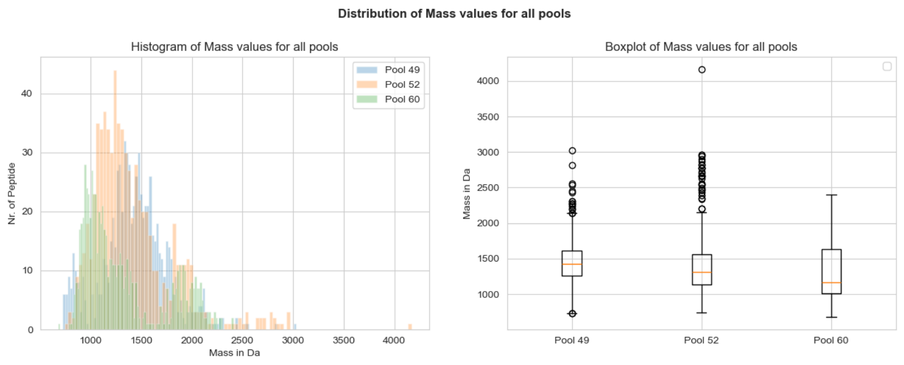

Figure 6.4: Distribution of all three pools. According to the authors of the dataset, all pools should show a similar distribution of the mass values. While Pool 49 and 52 show a comparable spread, Pool 60 exhibits strong differences. This could a possible explanation for the better performance associated with Pool 60.

SEQ attribute. In the case of this comparative analysis, the sequence “PEPTIDE” was used. By default, the algorithm excludes spectra where the absolute value of the difference between the peptide mass and the mass of the provided sequence are greater than 1000. The use of longer or heavier sequences in the SEQ attribute, leads to exclusion of fewer spectra. Although this part of the code is marked for training-use only, it is still executed when the analysis is started, as described in 3.1.2. Looking at the first 4000 spectra of Pool 60, 90 spectra were excluded when using “SEQ=PEPTIDE”, but zero spectra were excluded when using “SEQ=WWWWWW”. At best, the authors made a mistake when providing the instructions for correct usage. In the worst case, this is a mistake in the source code, and should be corrected.

6.2.3 Number of Peaks for DirecTag processing

DirecTag keeps the N peaks with the highest intensity rank. N can be set by the user, however, SearchGUI did not allow for the default value of 100 to be changed. These top N peaks are subsequently used to construct a spectrum graph. Peaks in a spectrum are seen as nodes. Two nodes are linked by an edge, if the corresponding mass difference relates to the mass of an amino acid. [31] Since there are spectra that DirecTag could not provide a single tag for, and in the light of the peak selection, the relationship between the available number of peaks and DirecTag’s predictions was examined. A low number of peaks does not automatically mean that they are of low quality. But the peaks that are present could not be as representative of the analysed peptide as expected. Figure 6.5 shows that the spectra that DirecTag could not predict any tags for, usually show a lower number of peaks.

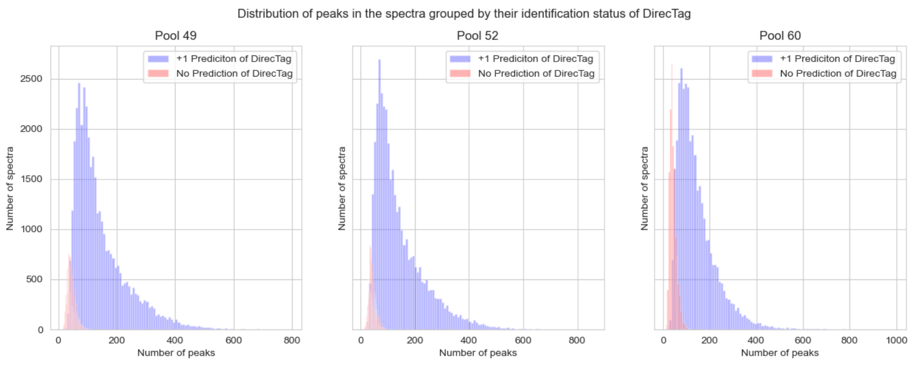

Figure 6.5: Distribution of number of peaks for spectra with no predicted tag (red) and spectra with at least one predicted tag (blue). Spectra with no tag have a lower number of peaks. These peaks could not be representative of the original peptide, and thus DirecTag cannot predict a tag for them.

6.3 Weaknesses and Limitations of the Method

6.3.1 DirecTag

By averaging the tags for one spectrum, DirecTag could be integrated in the analysis. Still, the fact that only sequence tags are produced was a bit limiting. Associating correct tags with MaxQuant results gives insight about how good DirecTag can perform on a spectrum level. However, linking the tags to sequences from the inclusion list is not fair towards the other three algorithms. The sequence tags used in this analysis exhibit a length of four amino acids. It is highly likely that more than one sequence of the inclusion list has a substring with 100% similarity to a given sequence tag. As a direct consequence, this association does not speak for correct identification of the spectrum. This fact is shown in the unprecedented boost of identifications in DirecTag when using the percent similarity score.

Chapter 7

Conclusion

7.1 Summary of Findings

The four selected algorithms give some good insights into the capabilities of de novo sequencing. Unfortunately, none of the four was able to beat the performance of database search in the form of MaxQuant results.

Based on the information provided in the results and discussion parts, the best performing algorithm can be named. Due to possible explanations stated above, the algorithms DeepNovo and DirecTag did not perform as well as the other two. Both, Novor and PEAKS were able to identify 38% of all spectra according to the percent similarity metric. Additionally, most of their predictions are in line with the actual sequences in terms of Levenshtein distance and normalized local alignment score. Nevertheless, Novor is to be preferred over PEAKS. Novor covered 99% of all spectra in terms of predictions. Furthermore, the score that Novor provides is more coherent with the correct identification in terms of identity and similarity.

7.2 Outlook

Since all algorithms were used with their default parameters, no optimizations were carried out in this analysis. This includes the use of a pretrained model in DeepNovo. Further analysis can be made by fine-tuning the algorithms. Training of a custom model might empower DeepNovo to achieve higher identification results than Novor/PEAKS or even MaxQuant. Investigation of stability can be expanded. Some algorithms use stochastic elements (e.g., simulation during DirecTag startup), that might lead to different results. Moreover, datasets with different fragmentation methods can be used.

Appendix A

Acronyms

MS	Mass Spectrometry
MS/MS	Tandem Mass Spectrometry
MALDI	Matrix-assisted Laser Desorption/Ionisation
ESI	Electrospray Ionization
CID	Collision-Induced-Dissociation
ETD	Electron Transfer Dissociation
PTM	Post-Translational Modification
HPLC	High Pressure Liquid Chromatography
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
JRE	Java Runtime Environment
SSE	Summed Squared Error
HCD	Higher-energy Collisional Dissociation
IQR	interquartile range

References

Literature

[1] Thilo Muth and Bernhard Y Renard. “Evaluating de novo sequencing in proteomics: already an accurate alternative to database-driven peptide identification?” *Briefings in Bioinformatics* 19.5 (Mar. 2017), pp. 954–970. DOI: 10.1093/bib/bbx033 (cit. on p. 1).

[2] Paul J Jannetto and Robert L Fitzgerald. “Effective Use of Mass Spectrometry in the Clinical Laboratory”. *Clinical Chemistry* 62.1 (Jan. 2016), pp. 92–98. DOI: 10.1373/clinchem.2015.248146 (cit. on p. 2).

[3] Xuemei Han, Aaron Aslanian, and John R Yates. “Mass spectrometry for proteomics”. *Current Opinion in Chemical Biology* 12.5 (Oct. 2008), pp. 483–490. DOI: 10.1016/j.cbpa.2008.07.024 (cit. on p. 2).

[4] Ruedi Aebersold and Matthias Mann. “Mass spectrometry-based proteomics”. *Nature* 422.6928 (Mar. 2003), pp. 198–207. DOI: 10.1038/nature01511 (cit. on pp. 2, 4).

[5] Anas El-Aneed, Aljandro Cohen, and Joseph Banoub. “Mass Spectrometry, Review of the Basics: Electrospray, MALDI, and Commonly Used Mass Analyzers”. *Applied Spectroscopy Reviews* 44.3 (Apr. 2009), pp. 210–230. DOI: 10.1080/05704920902717872 (cit. on pp. 3, 4).

[6] Sharad Medhe. “Mass Spectrometry: Detectors Review”. *Chemical and Biomolecular Engineering* 3.4 (2018), pp. 51–58. DOI: 10.11648/j.cbe.20180304.11. eprint: https://download.sciencepg.com/pdf/10.11648.j.cbe.20180304.11 (cit. on p. 3).

[7] Eshita Garg and Muhammad Zubair. “Mass Spectrometer”. eng. In: *StatPearls*. Treasure Island (FL): StatPearls Publishing, 2024. URL: http://www.ncbi.nlm.nih.gov/books/NBK589702/ (visited on 05/12/2024) (cit. on p. 3).

[8] Vicki H. Wysocki et al. “Mass spectrometry of peptides and proteins”. *Methods* 35.3 (Mar. 2005), pp. 211–222. DOI: 10.1016/j.ymeth.2004.08.013 (cit. on p. 4).

[9] Min-Sik Kim and Akhilesh Pandey. “Electron transfer dissociation mass spectrometry in proteomics”. *PROTEOMICS* 12.4–5 (Jan. 2012), pp. 530–542. DOI: 10.1002/pmic.201100517 (cit. on p. 4).

[10] Yaoyang Zhang et al. “Protein Analysis by Shotgun/Bottom-up Proteomics”. *Chemical Reviews* 113.4 (Feb. 2013), pp. 2343–2394. DOI: 10.1021/cr3003533 (cit. on p. 4).

[11] Teodor Octavian Nicolescu. "Interpretation of Mass Spectra". In: *Mass Spectrometry.* InTech, June 2017. DOI: 10.5772/intechopen.68595 (cit. on p. 4).

[12] Eric W. Deutsch. "File Formats Commonly Used in Mass Spectrometry Proteomics". *Molecular amp; Cellular Proteomics* 11.12 (Dec. 2012), pp. 1612–1621. DOI: 10.1074/mcp.r112.019695 (cit. on p. 5).

[13] Kevin Gurney. *An introduction to neural networks.* CRC press, 2018 (cit. on p. 6).

[14] AD Dongare, RR Kharde, Amit D Kachare, et al. "Introduction to artificial neural network". *International Journal of Engineering and Innovative Technology (IJEIT)* 2.1 (2012), pp. 189–194 (cit. on p. 6).

[15] Carlos Gershenson. "Artificial neural networks for beginners". *arXiv preprint cs/0308031* (2003) (cit. on p. 6).

[16] Keiron O'shea and Ryan Nash. "An introduction to convolutional neural networks". *arXiv preprint arXiv:1511.08458* (2015) (cit. on p. 6).

[17] Waseem Rawat and Zenghui Wang. "Deep convolutional neural networks for image classification: A comprehensive review". *Neural computation* 29.9 (2017), pp. 2352–2449 (cit. on p. 6).

[18] Hojjat Salehinejad et al. "Recent advances in recurrent neural networks". *arXiv preprint arXiv:1801.01078* (2017) (cit. on pp. 6, 7).

[19] Robin M Schmidt. "Recurrent neural networks (rnns): A gentle introduction and overview". *arXiv preprint arXiv:1912.05911* (2019) (cit. on p. 6).

[20] Yong Yu et al. "A review of recurrent neural networks: LSTM cells and network architectures". *Neural computation* 31.7 (2019), pp. 1235–1270 (cit. on p. 7).

[21] Carl Kingsford and Steven L Salzberg. "What are decision trees?" *Nature biotechnology* 26.9 (Sept. 2008), pp. 1011–1013. DOI: 10.1038/nbt0908-1011. (Visited on 05/13/2024) (cit. on p. 7).

[22] Yan-yan SONG and Ying LU. "Decision tree methods: applications for classification and prediction". *Shanghai Archives of Psychiatry* 27.2 (Apr. 2015), pp. 130–135. DOI: 10.11919/j.issn.1002-0829.215044. (Visited on 05/13/2024) (cit. on p. 7).

[23] Nabil Buhisi and Samy Abu-Naser. "Dynamic Programming as a Tool of Decision Supporting". *Journal of Applied Sciences Research* 5 (June 2009), pp. 671–676 (cit. on p. 7).

[24] Ngoc Hieu Tran et al. "De novo peptide sequencing by deep learning". *Proceedings of the National Academy of Sciences* 114.31 (July 2017), pp. 8247–8252. DOI: 10.1073/pnas.1705691114 (cit. on pp. 8, 9).

[25] Bin Ma et al. "PEAKS: powerful software for peptide de novo sequencing by tandem mass spectrometry". *Rapid Communications in Mass Spectrometry* 17.20 (Sept. 2003), pp. 2337–2342. DOI: 10.1002/rcm.1196 (cit. on pp. 8, 12).

[26] Bin Ma. "Novor: Real-Time Peptide de Novo Sequencing Software". *Journal of the American Society for Mass Spectrometry* 26.11 (June 2015), pp. 1885–1894. DOI: 10.1007/s13361-015-1204-0 (cit. on pp. 8, 14).

[27] Ari Frank and Pavel Pevzner. “PepNovo: de novo peptide sequencing via probabilistic network modeling”. eng. *Analytical Chemistry* 77.4 (Feb. 2005), pp. 964–973. DOI: 10.1021/ac048788h (cit. on p. 8).

[28] Hieu Tran. *nh2tran/DeepNovo*. original-date: 2017-06-22T13:47:50Z. Apr. 2024. URL: https://github.com/nh2tran/DeepNovo (visited on 05/03/2024) (cit. on pp. 10, 38).

[29] *cython/cython*. original-date: 2010-11-21T07:44:20Z. May 2024. URL: https://github.com/cython/cython (visited on 05/03/2024) (cit. on p. 11).

[30] Lijuan Mo et al. “MSNovo: A Dynamic Programming Algorithm for de Novo Peptide Sequencing via Tandem Mass Spectrometry”. *Analytical Chemistry* 79.13 (June 2007), pp. 4870–4878. DOI: 10.1021/ac070039n (cit. on p. 14).

[31] David L. Tabb et al. “DirecTag: Accurate Sequence Tags from Peptide MS/MS through Statistical Scoring”. *Journal of Proteome Research* 7.9 (July 2008), pp. 3838–3846. DOI: 10.1021/pr800154p (cit. on pp. 16, 39).

[32] Q. Li et al. “Fisher’s method of combining dependent statistics using generalizations of the gamma distribution with applications to genetic pleiotropic associations”. *Biostatistics* 15.2 (Oct. 2013), pp. 284–295. DOI: 10.1093/biostatistics/kxt045 (cit. on p. 16).

[33] Daniel P Zolg et al. “Building ProteomeTools based on a complete synthetic human proteome”. *Nature Methods* 14.3 (Jan. 2017), pp. 259–262. DOI: 10.1038/nmeth.4153 (cit. on p. 18).

[34] Compomics. *ThermoRawFileParser*. en. URL: http://compomics.github.io/projects/ThermoRawFileParser (visited on 05/29/2024) (cit. on p. 19).

[35] Waqar Haque, Alex Aravind, and Bharath Reddy. “Pairwise sequence alignment algorithms: a survey”. In: *Proceedings of the 2009 conference on Information Science, Technology and Applications*. 2009, pp. 96–103 (cit. on p. 21).

[36] Lisa Mullan. “Pairwise sequence alignment—it’s all about us!” *Briefings in Bioinformatics* 7.1 (Mar. 2006), pp. 113–115. DOI: 10.1093/bib/bbk008 (cit. on p. 21).

[37] Rakesh Trivedi and Hampapathalu Adimurthy Nagarajaram. “Substitution scoring matrices for proteins - An overview”. *Protein Science* 29.11 (Oct. 2020), pp. 2150–2163. DOI: 10.1002/pro.3954 (cit. on p. 21).

[38] Alex CW May. “Percent sequence identity: the need to be explicit”. *Structure* 12.5 (2004), pp. 737–738 (cit. on p. 22).

[39] Bas E. Dutilh. *SystemsBiology:BioinformaticDataAnalysis*. Lecture slides. Feb. 2016. URL: https://tbb.bio.uu.nl/BDA/2017/20170221_quantifying_sequence_similarity.pdf (cit. on p. 23).

[40] Bonnie Berger, Michael S Waterman, and Yun William Yu. “Levenshtein distance, sequence comparison and biological database search”. *IEEE transactions on information theory* 67.6 (2020), pp. 3287–3294 (cit. on p. 23).

[41] Executable Books Community. *Jupyter Book*. 2020. DOI: 10.5281/ZENODO.4539666 (cit. on p. 24).

[42] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual.* Scotts Valley, CA: CreateSpace, 2009 (cit. on p. 24).

[43] J. D. Hunter. “Matplotlib: A 2D graphics environment”. *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55 (cit. on p. 24).

[44] Michael L. Waskom. “seaborn: statistical data visualization”. *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021 (cit. on p. 24).

[45] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-06 86-2 (cit. on p. 24).

Software

[46] Lukas Steininger. *Comparative Work on De Novo Mass Spectrometry Algorithms.* URL: https://github.com/LukasStonie/PeptideDeNovoSequencing (cit. on pp. vi, vii, 25).

[47] Bioinformatics Solutions Inc. 2023. *PEAKS DeepNovo.* en-CA. URL: https://ww w.bioinfor.com/peaks-deepnovo/ (visited on 04/02/2024) (cit. on pp. 12, 13).

[48] CompOmics. *searchgui.* en. URL: http://compomics.github.io/projects/searchgui (visited on 05/03/2024) (cit. on p. 15).

[49] The pandas development team. *pandas-dev/pandas: Pandas.* Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134 (cit. on pp. 20, 21, 24).

[50] M. Bachmann. *Levenshtein: Python extension for computing string edit distances and similarities.* URL: https://github.com/rapidfuzz/Levenshtein (visited on 05/28/2024) (cit. on p. 23).

Online sources

[51] Alshaver. *Example mass spectrum. Mass to charge ratio on x-axis. Relative abundance on y-axis.* This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit http://creativecom mons.org/licenses/by/4.0/. 2016. URL: https://commons.wikimedia.org/wiki/File :Masspectrum.jpg (cit. on p. 5).

[52] Bioinformatics Solutions Inc. 2023. *PEAKS DeepNovo.* 2023. URL: https://www.b ioinfor.com/peaks-deepnovo/ (cit. on p. 12).

Check Final Print Size

— Check final print size! —

width = 100mm
height = 50mm

— Remove this page after printing! —