



**Universiteit
Leiden**
The Netherlands

Opleiding Informatica

Evaluating Protein Structure Prediction Algorithms

Donovan Jaeger

Supervisors:

Dr. Erik Schultes & Dr. Ir. Fons Verbeek

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)

www.liacs.leidenuniv.nl

09/10/2017

Abstract

Proteins are linear chains of amino acids found throughout life on earth. Proteins derive their diverse biological functions by acquiring complex folded structures, that are determined by the ordering of the amino acid sequences. The Protein Folding Problem, predicting the folded conformation from amino acid sequence alone, is a Grand Challenge in computer science. Over the last 30 years, many machine learning applications have been developed to address the Protein Folding Problem. However, as these many independent research efforts tend to use different technologies, training sets, and performance indicators, a systematic evaluation of the algorithms has not been possible. Here, we develop a method by which a number of algorithms predicting the secondary structure of protein sequences can be validated and compared head-to-head. First, we defined a universal representation for algorithm output accounting for Helix (H), Sheet (E) and Coil (L) structural elements. This enabled the development of metrics that compare the similarity between different protein structures. Universal structure representation and structure similarity metrics allow direct and meaningful comparisons of predicted structures. We used these methods to measure the performance of 4 algorithms predicting the secondary structure of protein sequences, using nearly 70,000 entries from the PDB (a large repository of experimentally determined protein sequence-structure information). From these results, we could then compare the performance of each of the 4 algorithms to the others. This method is comprehensive (using all known experimentally determined structure data) and universal (agnostic to the machine learning technology), and can be applied to any existing and potential secondary structure prediction algorithm.

Acknowledgements

I would like to thank Shamanou van Leeuwen for his help with the dataset, and Erik Schultes and my parents for their support, help and patience.

Contents

Acknowledgements	2
1 Introduction	1
1.1 The Gap Between Protein Sequence Data and Structure Data	1
1.2 Research Question	2
1.3 Thesis Overview	3
2 Methods And Background	4
2.1 Protein Sequence And Structure Data	4
2.1.1 MongoDB	5
2.2 Python	5
2.2.1 PyMongo	5
2.2.2 Jellyfish	6
2.2.3 Matplotlib	6
2.2.4 Pandas	6
2.3 The Algorithms	6
2.3.1 PsiPred	7
2.3.2 Jnet	7
2.3.3 Agadir	7
2.4 Experiment Workflow	7
3 Data: Preprocessing and Overview	9
3.1 Data Preprocessing	9
3.2 Data Overview	9
3.2.1 Unknown Elements	10
3.3 Experiment	13
3.3.1 Jaro Winkler Edit Distance	14
3.3.2 Matching Algorithm	14
4 Results and Discussion	16
4.1 Jaro-Winkler Edit Distance	16

4.2	Match Per Position Method	17
4.2.1	Agadir	17
4.2.2	Jnet	18
4.2.3	PsiPred	21
4.3	Summary	22
5	Discussion And Conclusions	23
5.1	Discussion	23
5.1.1	Incomplete Structure Data	23
5.1.2	Structure Similarity	23
5.1.3	Validation And Comparisons	24
5.2	Conclusion	24
	Bibliography	25

Chapter 1

Introduction

Proteins are large macromolecules that are fundamental to all our biological processes. All living beings are built by proteins [1]. Each protein consist of one or more amino acid chains and there are twenty amino acid found in living organisms. An amino acid chain consist of one or more amino acids. Proteins derive their diverse biological functions by folding these chains of amino acids. These folds and function are determined by the ordering of the amino acid sequences.

We can divide protein structure four levels [2]: primary structure, secondary structure, tertiary structure and quaternary structure. In this thesis we are only focussing on primary and secondary structure. Primary structure simply is the sequence of amino acids. When we refer to sequences in this paper, we refer to this primary structure. With Secondary structures we begin to see the formation of α -helix and β -sheet.

The 'Protein Folding Problem', predicting the structure from amino acid sequence alone, is a Grand Challenge in computer science [1].

1.1 The Gap Between Protein Sequence Data and Structure Data

The Protein Data Bank (the PDB) was established in 1971 as a computer based archive for macromolecular structures. Its purpose is to collect, standardize and distribute data from crystallographic studies such as solved protein and nucleic acid structures [3]. Starting out with only 7 structures archived, the number of deposited structures grew incrementally. This was due to the advances in technology in all aspects of the crystallographic process that are used to determine structure, new technologies to determine structure such as nuclear magnetic resonance (NMR) and changing views about sharing data [4]. Since 2015, over 100,000 structures have been archived in the PDB. Knowledge of three-dimensional structures of proteins are a necessity for understanding how it functions in the body and can help us to understand and treat diseases.

Although advances are being made in the speed by which three-dimensional structures are being determined,

there remains a widening gap between sequence information and corresponding structures. There are around 100,000 structures known to us, however, there are presently over 90 million sequences at our disposal [5]. In 2009 only 0.6% of the protein sequences in UniprotKB/TrEMBL had a solved protein structure in the PDB compared to 2% in 2004 and 1.2% in 2007 [6]. And with DNA sequencing technologies such as those used in the Human Genome Project, it means that knowledge of protein sequences will continue to outpace structural biology for the foreseeable future. As a result new methods must be developed to predict structural and functional features using only the amino acid sequences.

1.2 Research Question

Computational techniques used to predict protein secondary structure (classified in three states: α -helix, β -sheet and coil) have, over the last 30 years, improved in accuracy from around 50% to over 80% [5]. When these techniques were first conceived by their research and development groups, performances was tested on the available sequence-structure data (see Chapter 2 for more details on this). Since then the the number of known structures in the PDB has grown, meaning that it is now possible to test the algorithms against a greater number of sequences and associated structure.

In this thesis we use the much larger contemporary PDB resources to validate the performance of multiple protein structure prediction algorithms. We do this by comparing each protein structure predicted by each of these algorithms to the experimentally determined structures in the PDB. This will result in a benchmarking of each algorithm, such that the algorithms can be compared side by side. We formulate this approach in the following research question:

In general, how can the performance of protein structure prediction algorithms be systematically validated and compared?

In order to answer our research question we will investigate the following subquestions.

First, we need to know what data we are using for our experiments. For example: how many sequences and structures are represented in the dataset? Second, What metric is best used when comparing two structures with each other? Third, we are interested in the performances between the prediction algorithms: Is one algorithm performing better than the others? Lastly, in both the experimentally determined structures of the PDB and the computationally predicted structures from the algorithms we find amino acid positions that remain uncertain. This means that there are still a number of structural elements classified as unknown in the PDB as well as a number of unknown structural elements in the predictions. These residual undetermined positions can confound our interpretation of the results. This required carefull consideration when assessing structural similarities.

1.3 Thesis Overview

Now that we have our problem statement and research question defined we can continue with our thesis. First, in Chapter 2 we discuss our methods and tools used for our experiments and give more information about protein structures. In Chapter 3 we discuss the preprocessing performed on our dataset and give more of an overview on the data we have used. Then, in Chapter 4 we discuss the results of our experiments. Finally, in Chapter 5 we conclude our work.

This document is a bachelor thesis for the 'Computer Science & Economics' program at the LIACS of Leiden university. The project is supervised by Dr. Erik Schultes and Dr. Ir. Fons Verbeek.

Chapter 2

Methods And Background

2.1 Protein Sequence And Structure Data

The dataset used was originally derived by a previous investigation (S. Van Leeuwen & E. Schultes, personal communication, 2017-04-15.) The dataset consists of three MongoDB collections (equivalent to tables in SQL) as seen in figure 2.1. These three collections are 'algorithms', 'sequences' and 'structures'. This dataset contained 69,058 protein sequences procured from the PDB, their experimentally determined structures (target structure) and the structures predicted by the various algorithms (predicted structure).

The 'algorithm' collection consisted of four columns, and four entries (algorithms). The information it contained are the names of the algorithms and its version number that is used for this experiment. There is a citation to the method and algorithm and an unique ID. The algorithms used for this collection are: Jnet [7], Agadir [8], PsiPred [9] and PSSpred [10]. However, it was necessary to exclude PSSpred from our experiment as it failed to produce interpretable results. All confidence values reported by PSSpred were 0.0, meaning that all predictions were essentially random. This null result remains unexplained so we excluded it from our analysis.

The 'sequences' collection consisted of seven columns. Each entry has values for its name which becomes its unique ID and is used as foreign key in the 'structures' collection. The sequence value contains the protein sequence itself and the pdb ID value contains a reference which you could use to find the respective sequence in the PDB itself.

The largest collection is the 'structures' collection having 175,871 entries. Besides its own unique ID, it also has a reference to the IDs of the algorithms (algorithmHash) and sequence (sequenceHash) to describe which algorithm is used to get this predicted structure and the source of the protein sequence. The 'shorthandOutput' gives us the structure as a string separated with a space between each character. The characters display a structural motif and are either H (alpha Helix), E (beta Sheet), L (coil) or N (unknown). The 'output' also gives us the structure but is in this case augmented with the level of confidence for each character. This confidence value shows the algorithm's level of confidence in its prediction of that structural element, and is between 1.0

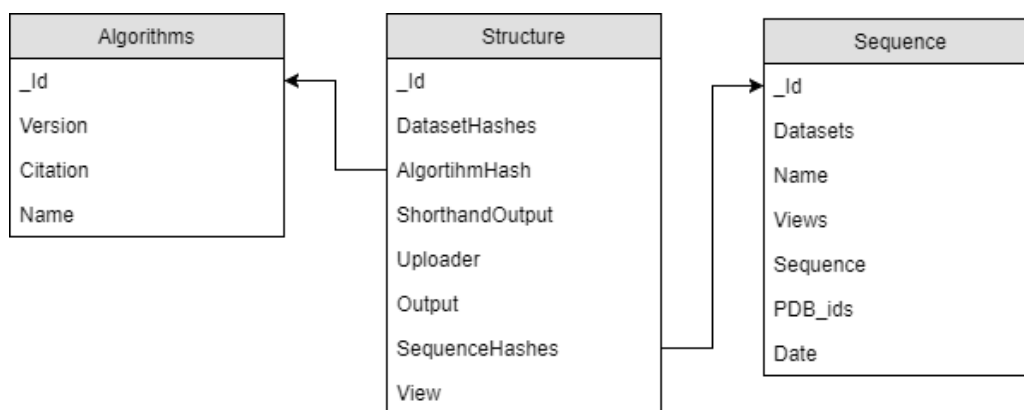


Figure 2.1: The database model with the three collections: Algorithms, Structure and Sequence.

(absolute certainty) and 0.0 (completely uncertain).

2.1.1 MongoDB

The data used for this research is stored in MongoDB [11], a NoSQL database. In contrast to relational database management systems such as SQL, entries do not have to abide by a set structure and are schemaless. Entries, or documents, can have a dynamic structure; meaning that not every document has the same field and therefore does not have to hold the same kind of information. It also means that entries do not affect each other, and new data can easily be added even if the database is already very large [12]. All this makes MongoDB a much more flexible database system. The version used for this project is version 3.4.3.

2.2 Python

To interpret the data we used Python as our programming language to write our scripts. Python supports many libraries which makes it easier to develop useful models to interpret the data. Python is very flexible with regard to memory allocation compared to other well-known languages which make it very powerful for a data analysis. Releases are also well documented by the developers at the Python Software Foundation [13]. The version we used is 3.5.2 although the latest release at the moment of writing is 3.6.2.

2.2.1 PyMongo

One of the major Python libraries that we used during our experiments is PyMongo. PyMongo allows us to interact with the database in MongoDB and to perform queries to search for certain parts of the database. In our experience it is faster and more flexible than other Python/MongoDB libraries such as MongoEngine and is the recommended library by the MongoDB developers team [14].

2.2.2 Jellyfish

Jellyfish [15] is another Python library that we used for our experiments. Jellyfish offers many different functions for string comparisons. It allows us to compare the different structures to each other. To compare the structures we are going to make use of a function called the 'edit distance'. The edit distance is a model that calculates the minimum amount of operations (either insertions, deletions or substitutions) needed to make one string identical to another one [16]. The edit distance is often used in spelling checkers [17] We will take a look at the following example where we want to determine the edit distance between JOB and BOBBY:

JOB → BOB → BOBB → BOBBY

The distance in this example would be 3 operations. There are different algorithms that use the principles of the edit distance such as the Damerau-Levenshtein [18] distance and the Jaro-Winkler distance [19], which can do transpositions as an extra operation. Being able to do transpositions is one we chose the Jaro-Winkler edit distance. The advantage of being able to swap adjacent characters is that it can correct misplaced characters and does not completely dismiss strings if the order is wrong but has the same composition of character as the target string. Another reason why we choose the Jaro-Winkler distance is because it normalizes the outcome between 0.0 (no similarity between two strings at all) and 1.0 (the two strings are identical) while other edit distances mostly return a total amount of edits needed regardless of the length.

2.2.3 Matplotlib

Matplotlib is a Python library which can plot 2D publication-quality figures. This library was used to provide us with the ability to visualize our results as it is the easiest and most distinct way to represent the large amounts of data that we have.

2.2.4 Pandas

Pandas is a Python library that provides high-performance, easy-to-use data structures and data analysis tools [20]. It helps us to sort and manage our data.

2.3 The Algorithms

We will now discuss the three¹ algorithms used to get our predicted structures.

¹We excluded PSSPred from our experiments in its entirety as the results were unreliable

2.3.1 PsiPred

PsiPred is using a two feed-forward neural networks to analyse data obtained from PSIBLAST². PSIBLAST is an algorithm that searches for similarities in parts of the sequence with other protein sequences which it then links with other sequences that are similar to those [21]. This process can repeat itself multiple times depending on how many iterations are desired. Finally, it uses consensus techniques to determine the final secondary structure more accurately using those alignment techniques. PsiPred is able to classify the three structural elements (H,E and L) but does not classify elements as unknown (N). The developers state that the algorithm has an average accuracy of 81.6% although they do not state how big their test set was. During CASP4³, a meeting evaluating and discussing protein structure prediction techniques, PsiPred had an average of 80.6% correctly predicted peptides over 40 sequences. As they state themselves, this is a small sample and is not statistically significant [22].

2.3.2 Jnet

Jnet is a prediction algorithm using a neural network [23] that works by applying multiple methods of aligning protein sequences [7] such as PSIBLAST. Jnet classifies all four structural elements (H,E, L and N). Jnet is tested using a testset of 406 non-redundant proteins apart of the training set. Structures predicted have an average (Q3) accuracy of 84%, in 68% of the cases.

2.3.3 Agadir

Agadir classifies only two structural elements (H and N). Therefore, Agadir is not considered as a algorithm to predict the secondary structure of proteins [24] but we use it in our experiments anyways to evaluate its performance of what it can do. The developers state that the algorithm is tested against 1,200 peptides to evaluate its performance which resulted in a standard deviation of $\sigma = 6\%$

2.4 Experiment Workflow

In figure 2.2 a workflow of our experiment is given. First we extracted 69,058 sequences and their experimental determined secondary structure (SS). Then, we used those sequences to predict SS with four structure prediction algorithms. We used the experimental SS as our target structure to compute a structure distance with the predicted SS. Finally we validated and compared our results from this experiment.

Next, we will discuss the preprocessing we did on the data and the data itself

²Position-Specific Iterated Basic Local Alignment Search Tool

³Fourth edition of the Critical Assessment of techniques for protein Structure Prediction

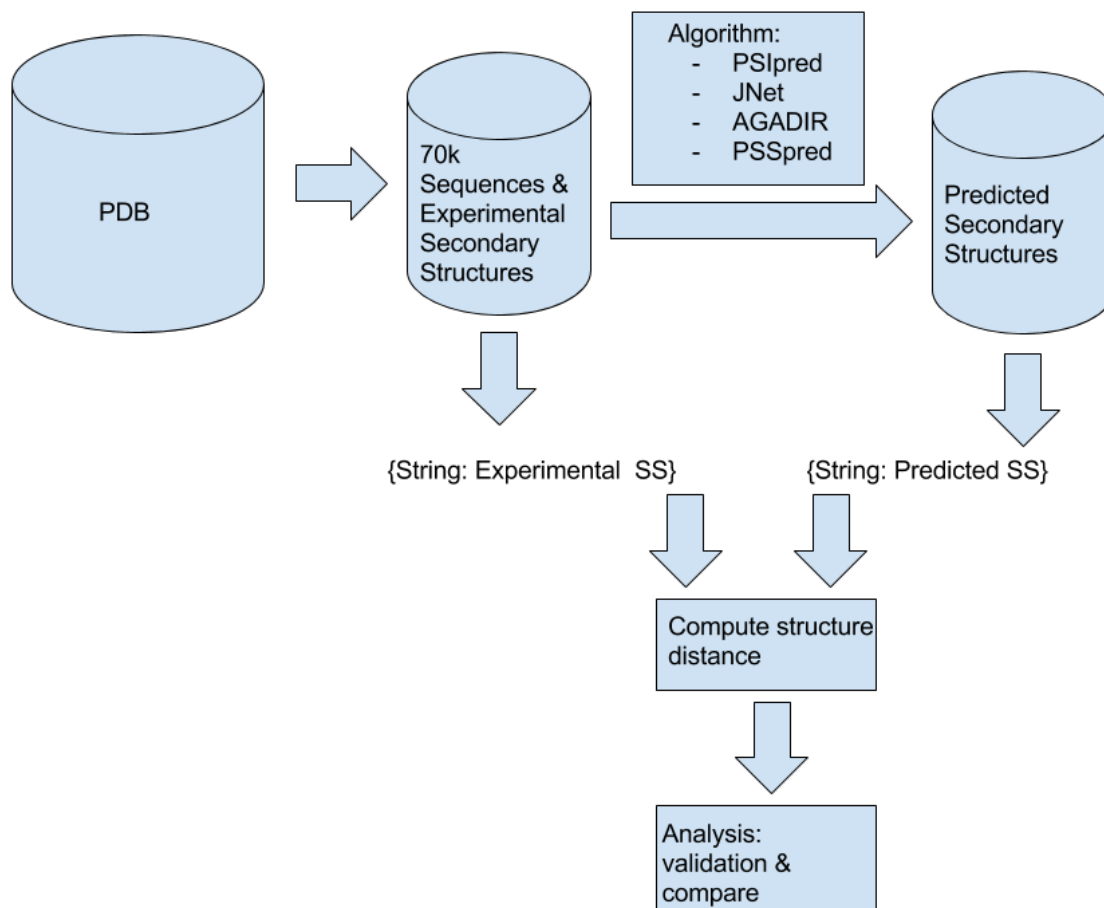


Figure 2.2: A comprehensive and universal method by which a number of algorithms predicting the secondary structure of protein sequences can be validated and compared head-to-head.

Chapter 3

Data: Preprocessing and Overview

3.1 Data Preprocessing

The goal of this research project is to validate the quality of protein prediction algorithms. We have taken 69,058 sequences with their experimentally determined secondary structure from the PDB and ran those sequences against three independently engineered secondary structure prediction algorithms.

We created a Python script that runs through each sequence and pairs the predicted structures with their respective target structure from the PDB. We used the indexes found in the sequence files to find the target structure and the corresponding predicted structures. We then put them in separate files for each of the corresponding algorithms the predicted structure was based on. For completeness, we tried to save as much information into these files so that we did not have to perform our processing code again in case we wanted more information as well as in case someone wants to work upon our data. The script then performs the Jaro-Winkler distance algorithm to compare the paired strings and gives back a score from 0.0 to 1.0 for each pairing. The higher the score, the more similar the two strings. Similar, if the score is low then the two strings are very different. The score is normalized for differences in sequence length. Table 3.1 shows us the structure of the files created by our Python script that extracted the data from the MongoDB database.

3.2 Data Overview

Figure 3.1 depicts the proportions of the predicted structures in the dataset. As can be seen, the three datasets are not equal in size, meaning that not all sequences are run through all three algorithms. For example, Psipred predicted structures for only 19,670 out of the total of 69,058 sequences available in this experiment. Furthermore, in total only 24,113 structures were predicted by all three prediction algorithms. The exact reason on why so many structures are missing is unknown to us. But it may be possible that it is the result of an error

Table Name	Description
sequenceHash	The hashkey used to reference the sequence in the database. Makes it easier to look up sequences afterwards.
sequence	The amino acid sequence with length N corresponding with the target and predicted structure.
PDB Structure	The target structure as found in the PDB
H.PDB	The number of α -Helix in the target structure.
E.PDB	The number of β -Sheet in the target structure.
L.PDB	The number of Coil in the target structure.
N.PDB	The number of positions with unknown structure.
predicted Structure	The predicted structure as predicted by one of the four algorithms
H_predicted	The number of α -Helix in the predicted structure.
E_predicted	The number of β -Sheet in the predicted structure.
L_predicted	The number of Coil in the predicted structure.
N_predicted	The number of positions with unknown structure
Structure similarity Score	The Jaro-Winkler distance between the Target structure and the predicted structure
Average of Confidence	$\frac{\sum_{i=1}^n \text{Confidence level}_i}{n}$ The average of all the confidence levels for this predicted structure
Length of sequence	Sequence length n
Length of PDB	Number of characters in the target structure. Should be equal to n
Length of prediction	Number of characters in the predicted structure. Should be equal to n

Table 3.1: The file structure of the extracted dataset.

in the batch code used to run all the sequences on the prediction algorithms. Some sequences may have failed on one prediction algorithm while it may have run correctly on the others.

Figure 3.2 shows us the union between the three sets formed by the predicted structures that were computed the three algorithms: $Agadir \cup Jnet \cup PsiPred$, as well as the intersection between the three algorithms. If we add up the numbers per algorithm (16,937) we notice that they are not the same as the total amount of structures we stated earlier (24,113). In Figure 3.2 we excluded duplicates and sequences that did not predict structures that are of the same length of the target structure. An explanation for the duplicates is that there were duplicates found in the collection of structures in the PDB, which source sequence we then use multiple times. It must be noted, however, that although the same sequences may be used multiple times, it does not necessarily mean that they produced the exact same structure. It may be the case that various experimental methods were used, possibly resulting in different structures. That is why we still use all 23,700 predicted structures available to us after we removed the structures that were not of the same length as their target structure as this would result in issues on how to align these structures correctly if they are not of the same length.

Let's look at $Agadir \setminus (Jnet \cup PsiPred)$ (sequences of Agadir without the sequences shared by Jnet and PsiPred) and $PsiPred \setminus (Jnet \cup Agadir)$ (sequences of PsiPred without the sequences shared by Jnet and Agadir). It is notable that the sets of sequences that are exclusive to that algorithm are bigger than the set of sequences that they both have predicted. In other words; there is not as much overlaps as we would hope for our experiment¹.

3.2.1 Unknown Elements

Next, we will look at the unknown elements in our target structures and predicted structures for each algorithm. Figures 3.3a-3.3c show us the average ratio of our structures that are unknown.

Figure 3.3a shows us the average ratio of N in the target structures and that of the structures predicted by

¹This may tell us that the algorithms had trouble predicting a certain sequence while other the algorithms did not have the same issue. Unfortunately we have no information of these failed sequences nor why they failed to execute

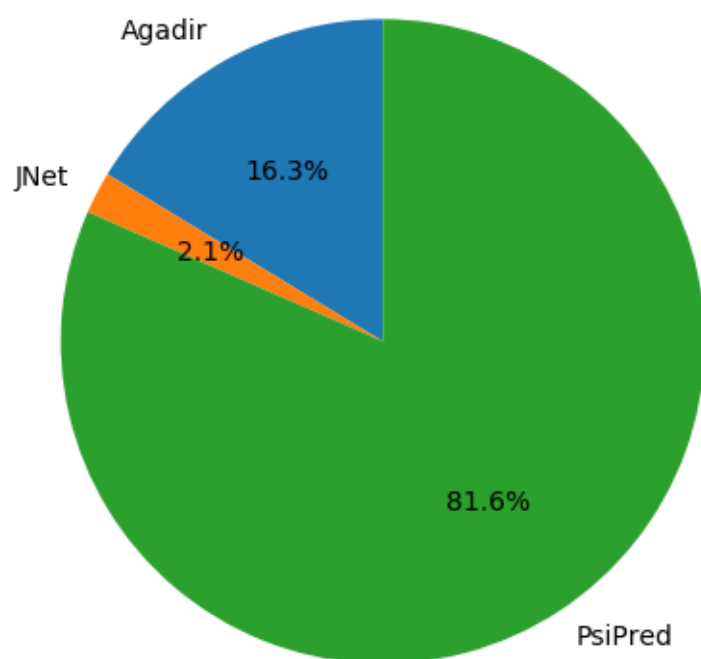


Figure 3.1: The proportions of computed structures per prediction algorithm in the dataset.

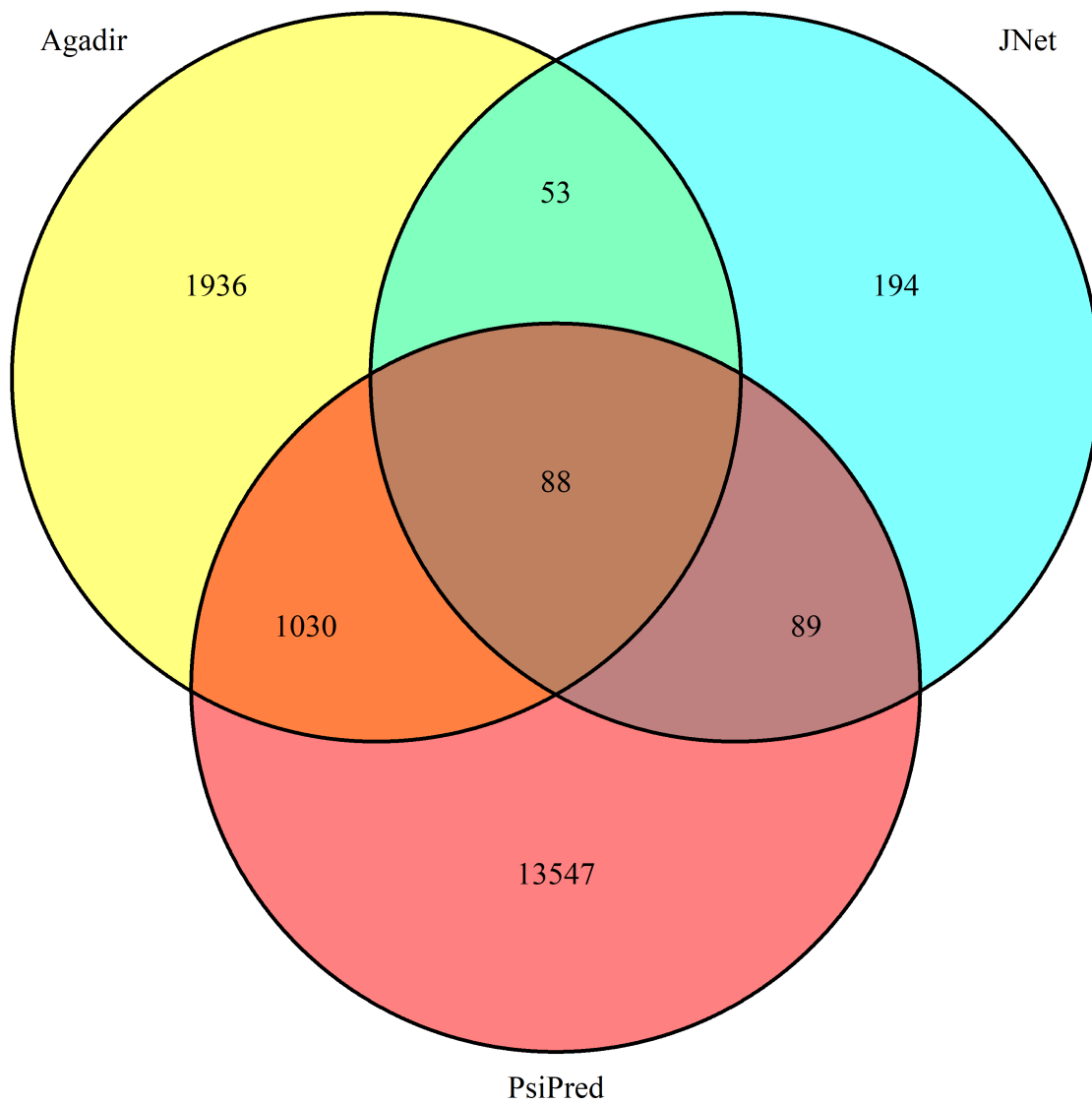


Figure 3.2: The overlaps between the sequences used by the prediction algorithms.

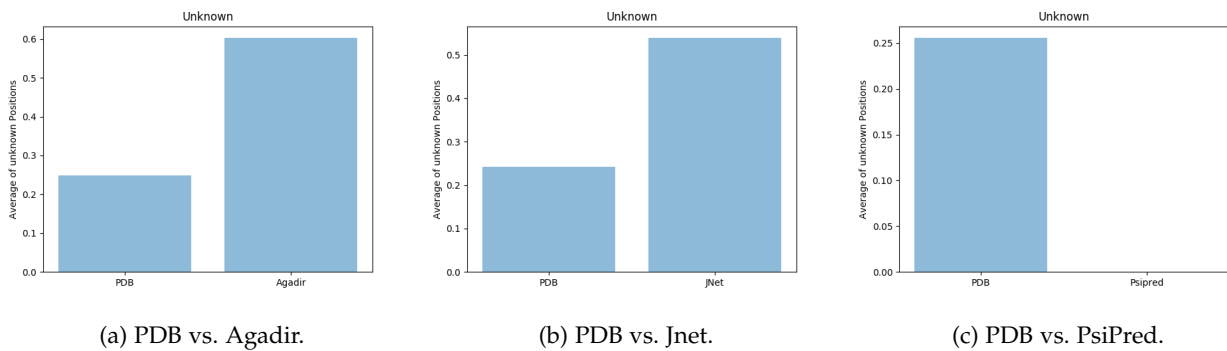


Figure 3.3: Ratios of unknown elements in the PDB structure and the predicted structures sorted by algorithm.

Agadir. We see an average of 0.2 in the PDB and 0.6 in our predicted structures. This seems like large number of unknown elements, but we have to consider that Agadir only predicts α -Helix. If the algorithm doesn't predict a certain element as Helical then it labels it as unknown.

Figure 3.3b shows us the average unknown element ratio from JNet and the target structures. The average of the target structure is 0.2 and the average of the predicted structure of JNet is 0.5. This is on average again over a half of all structure elements while JNet is supposed to be able to predict all three structural elements.

Lastly, we take a look at figure 3.3c. PsiPred gives us an average of 0.255 ratio on the target structures while there are no unknown structure elements in the predicted structure at all. This seems very logical as PsiPred does not assign an unknown classification unto structural elements. Instead it assigns it to a coil even if it is not certain about the structure at all.

Looking at this data we can confirm that the target structures of the PDB have an average of around %25 of unknown structure elements. Meaning that, even if considered as the gold standard in protein structure information, the PDB has a lot of unknown or missing structural information. This underlines the importance to find new and better methods to get structural information especially computational methods. We also see a high ratio of unknown elements with Agadir and Jnet. Although understandable in the case of Agadir it can be worrisome with Jnet. On the other hand: at least it admits that it is not sure about the structure as opposed to PsiPred that assigns everything to Coil anyway.

3.3 Experiment

Now that we know about the data we are using we can discuss how we will use this data in our experiment. Given our research question, we want to compare the predicted structure with the PDB structure which we refer to as the target structure.

To compare the structures we will handle each structure as a string composed of the structural elements H, E and L. We will take two approaches to compare strings:

	1	2	3	4	5	6
Target	L	N	E	H	H	N
Prediction	N	N	N	H	H	H

Table 3.2: Example of comparing a pair of structures.

1. Use the edit distance between two structures.
2. Count the number of correctly predicted structural elements for each position.

3.3.1 Jaro Winkler Edit Distance

As we already discussed in the methods and in the preprocessing section we will use the Jaro-Winkler edit distance to compare how similar two strings are to each other as a measure of structural similarity. We performed the Jaro-Winkler algorithm on each pair of the target structure and the corresponding prediction and sorted them by algorithm. The frequency distribution of the 'Structure Similarity Scores' will then be computed and visualized in a bar chart. In the same chart we will plot distribution of 'Structure Similarity Scores' computed from an analogous dataset of randomized structures (a permuted string of each structure, preserving its biological composition but not sequence order). This will serve as a benchmark to prove if the prediction algorithms performs better than randomly generated structures.

Note that we use the Jaro-Winkler distance with the JNet and PsiPred structures but the Jaro distance for the Agadir structures. This is because of the composition of the Agadir structures and the fact that the Jaro-Winkler variant also performs swapping elements in addition to inserting and removing. As Agadir structures only consist of 'H' and 'N', using the Jaro-Winkler could result in unfair results. It could either result in a low score as it would have to insert a number of other elements ('E' and 'L') and to remove some 'N' to transform the predicted structure to the target structure. The similarity would be very low but Agadir itself can't do anything about those unknown elements while all the Helical elements could be assigned correctly. On the other hand, we also experimented with changing the structural element in the target structure to the same alphabet that Agadir knows. For this we changed all 'H' and 'L' elements to 'N'. This way we could compare them on even grounds. However, this resulted in another issue as Jaro-Winkler can swap elements for a lower cost than inserting and removing. This means that all structures can be easily edited from one to another if the string only consists of two characters which you can swap at little costs. Because of this we decided to perform the simpler Jaro-distance for Agadir, only able to remove and insert elements while we maintained the two character target structure for fair comparison.

3.3.2 Matching Algorithm

The second approach we take is to compare the structural element for each position to its counterpart in the other structure. If it is the same character we will count it as a match.

We will test this method three times under different conditions. (1) We will use this method but will exclude the unknown structure elements in the target structure. (2) We will use this method but will exclude the unknown structure elements in the predicted structure. (3) We will use this method but will leave out the unknown structure elements in both structures. The reason we use these three methods is because we believe it will tell us more about the impact of unknown elements in the PDB as well as in the predictions. We are interested in how many structural elements the algorithms can predict correctly and as long as even the PDB has unknown structural elements, we believe that it is not entirely fair to include them in our comparison.

$$d_{Algorithm,Sequence} = \frac{\sum \text{matches in structure}}{\text{Length of sequence} - \text{UnknownElements}}$$

We will clarify this method using an example in table 3.2. In all three cases *Length of sequence* = 6, and the amount of positions that match would be two (position 4 and 5). However, the value in the denominator, and thus the result, would be different.

In (1) the target sequence has two unknown elements at position 2 and 3. This results in $\frac{2}{6-2} = \frac{1}{2}$. In (2) we see three unknown elements in the predicted structure at positions 1,2 and 3. This results in $\frac{2}{6-3} = \frac{2}{3}$. Finally at (3), we have unknown elements at position 1,2,3 and 6². This gives us a result of $\frac{2}{6-4} = 1$.

²Note that we only count the number of positions where there are unknown elements, not the total amount of unknown elements in both structures as it would be redundant and possibly larger than the sequence length itself.

Chapter 4

Results and Discussion

4.1 Jaro-Winkler Edit Distance

In the first experiment we applied the Jaro distance algorithm on Agadir and the Jaro-Winkler distance algorithm to the other two algorithms, to calculate the edit distance between our predicted structures and the target structures. To validate this method we generated random structures to act as a null distribution. These random structures are essentially the predicted structures but its elements are permuted. That way we get different structures while retaining the same composition of elements. This way we can benchmark if this method is accurate enough to compare two strings to each other where the exact position of an element is essential.

In the figure 4.1a we immediately notice that the edit distance of the structures predicted by Agadir is overall very good. We also see, however, that our null distribution performs similar to our predicted structures.

We see a similar trend with the other two algorithms in figures 4.1b and 4.1c. PsiPred structures in figure 4.1c are nearly identical to the null distribution.

Although the edit distance shows us that the predicted structures are in some degree similar to the target structures, they are no more similar than would be expected by chance. We expected the predicted and random graphs to be significantly different. One reason why this is not the case is because of the limited alphabet that these structures consist of. Jnet has four structural elements (H, E, L and N), while PsiPred has only three (H, E and L) and Agadir only two (H and N). This would make two strings very similar to each other by chance alone, especially if the compositions of these strings are the same. Because of this, there was a possibility that the random and predicted results would be the same, using the Jaro-winkler distance. Even after additional experimentation with the Jaro edit distance we could find no significant difference between the two distributions.

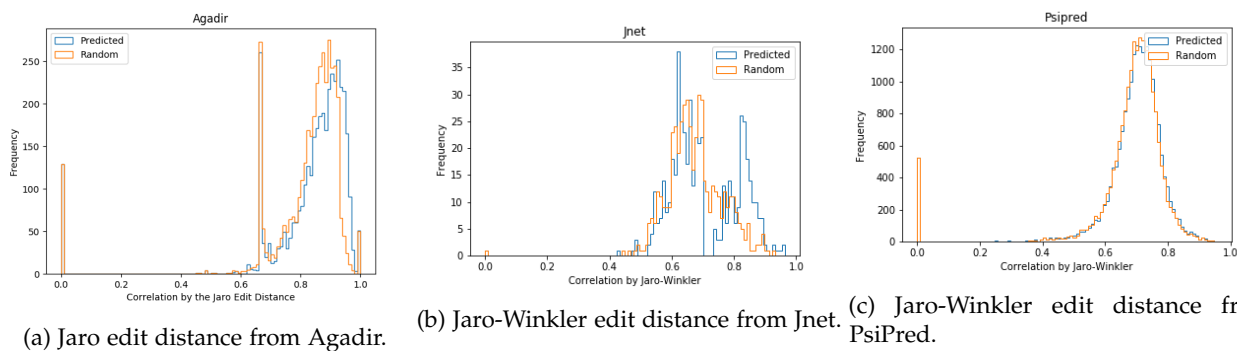


Figure 4.1: The frequency distribution of the edit distance between the two structures.

4.2 Match Per Position Method

Given that the Jaro-Winkler distance failed to separate the null and the predicted distribution, and that we assume that is related to a small alphabet (H,E and L) or large numbers of unknown structural elements (N), we devised an alternative metric. This alternative metric simply counts the number of matches per position between the predicted and the target structures, where unknown positions can be filtered out. Below we discuss the application of this metric for the three algorithms.

4.2.1 Agadir

We start of by discussing our results for Agadir. In figure 4.2 we show a scatterplot illustrating the number of matches that each predicted structure has with it corresponding target structure. The x -axis represent the length of the sequences while the y -axis represent the number of matches two structures have. The points each represent a sequence run through (in this case) Agadir and its respective structures.

We see that, although, the data points are generally upward-sloping it is highly scattered. We have to note, however, that this scatterplot may be unfair towards Agadir. As this plot is the number of matches against the length of the structure while Agadir only predicts Helical elements. That being said, it makes sense that the data is scattered.

Another thing that stands out is that the sequence length stops at 500. While, as you can see in figure 4.5 and 4.7, there are sequences longer than 500 with Jnet and PsiPred as well as with Agadir. The problem being that Agadir failed with sequences longer than 499. Agadir returned structures with $length = 0$ or $length = 1$ if $sequencelength \geq 500$. We excluded these cases as well as all other cases were $Target\ structure\ length \neq Predicted\ structure\ length$ for all our experiments in this section as we did not want to handle the problem of how to align structures with different length.

In figure 4.3a we see the ratio of matches and length of the sequence without the consideration of unknown structure elements in the target structure. We notice that, first of all, the predicted distribution lies largely on

the left side of the x -axis, meaning that there are not that many similarities between the predicted structure and the target structure. Secondly, the random distribution is just a small shift to the left in comparison to the predicted distribution. Overall, this does not seem very promising. This is, however, the reason why we used different conditions on how to calculate the ratio. As said earlier, Agadir only predicts Helical elements and classifies the rest as unknown. But as we only remove the number of unknown elements from the target structure, that means we still have a number of unknown elements in the predicted structure. Our method thus tries to compare e.g. a 'N' from the predicted structure, to an 'E' or an 'L' in the target which is not a match. So even if the predicted structure classified all helical elements correct, it will get a lower ratio score as Agadir can not predict β -sheet and coil.

Given this, in figure 4.3b we compute the matches without N in the predicted structures. We already see a bigger difference between the predicted plot and the random plot. Not only that, the most frequent cases also moved further to the right. When we further modify the match metric by filtering N in both the target and predicted structure we see, in figure 4.3c the mode of the predicted plot as well as the random plot have shifted a bit to the right.

Another salient point is that in each case there are more than 300 structures that have zero matches, or nearly zero matches, with the target structures. It stood out to us that there were so many predicted structures with no matches at all. Our observation on why this happens is that, first of all, there were a lot of sequences in the PDB that had no helical elements in it whatsoever. In that case there can not be any matches, obviously. Secondly, we noticed that in the cases there were helical elements, they were in smaller numbers than the other two elements, 'E' and 'L'. thirdly, in cases the helical elements were in a majority compared to the other two elements, they were in small numbers compared to the unknown elements. This is shown in figure 4.4, where the x -axis is represented by $Ratio = \frac{H}{H+E+L}$ and the y -axis by the same number of matches discussed in this section. In all three cases there is a clear upward-slope with some outliers as explained above.

4.2.2 Jnet

We continue with the same experiments on the structures predicted by Jnet. In figure 4.2 we see an upward relation between length of the sequence and the amount of matches between the prediction and target. We notice that there is a high concentration of structures until length = 250, which tells us that most of our sequences predicted by Jnet are shorter than 250 amino acids. After that the plot becomes less dense and with more deviation from the best fit line.

In figure 4.6a, showing the frequency distribution of matches without N in the target structure, we see that the null distribution and the prediction distribution are, once again, close to each other. This is the same case as with Agadir using the method where we neglect the unknown elements in the target structures. The predicted distribution has three peaks around 0.25, 0.4 and 0.55, while the null distribution peaks around 0.15. So while Jnet performs slightly better in this case, there remains large overlaps with the null distribution.

Figure 4.6b, which shows the frequency distribution of matches without N in the predicted structure, we

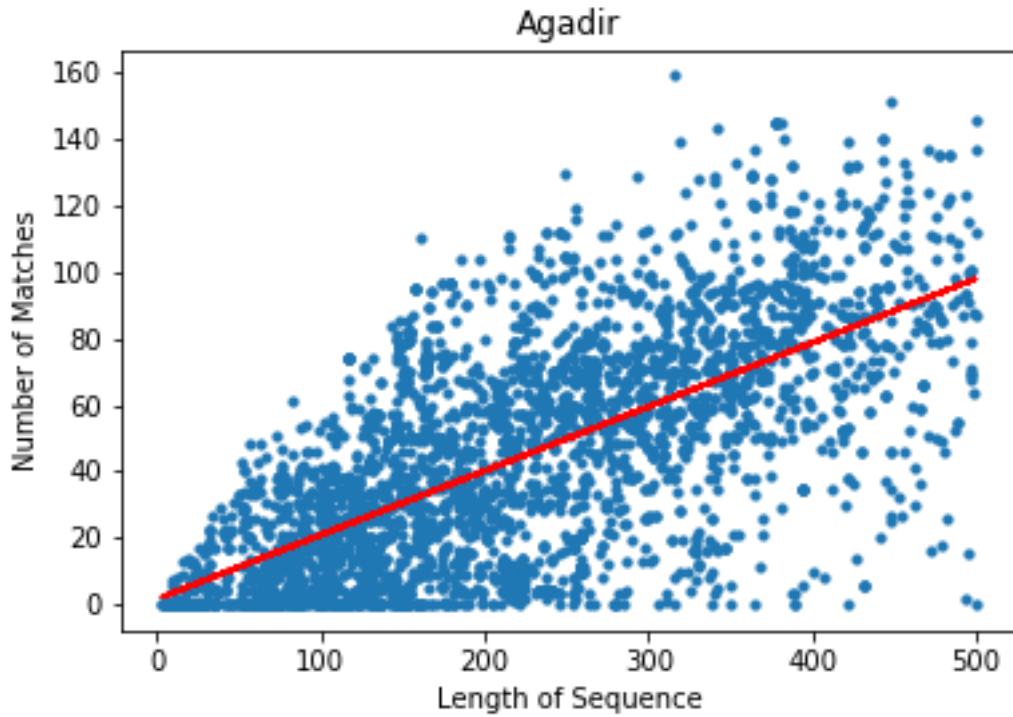


Figure 4.2: The relation between the length of a sequence and the number of correctly assigned structural elements by Agadir.

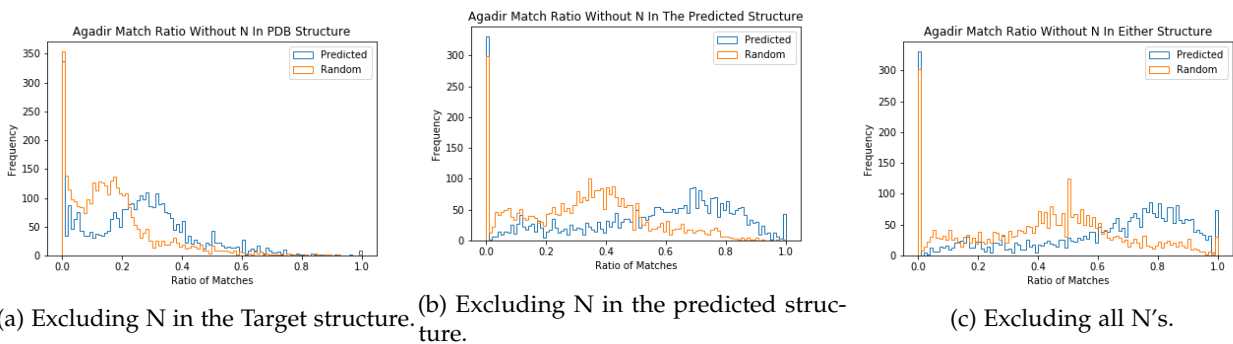


Figure 4.3: Frequency distributions of the ratios of matching elements with Agadir, using three different metric on how to handle unknown structural elements.

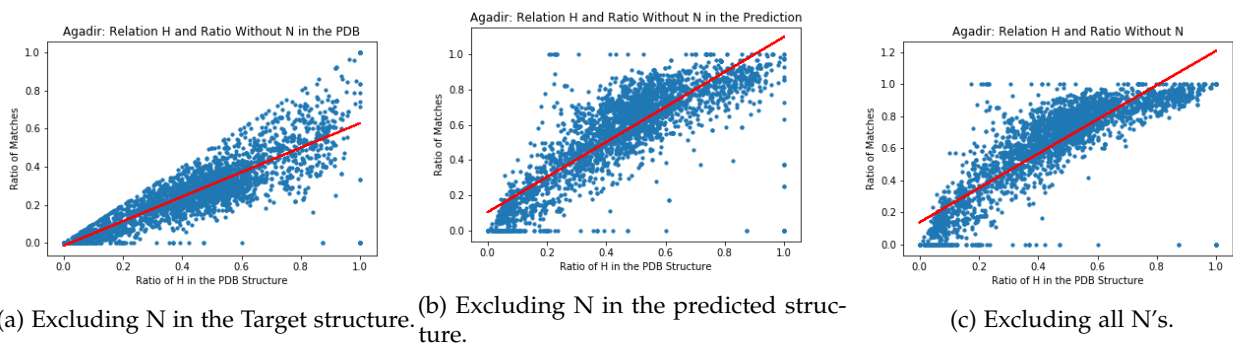


Figure 4.4: Scatterplots illustrating the relation between the ratio of Helical elements in the target structure and the ratio of matches.

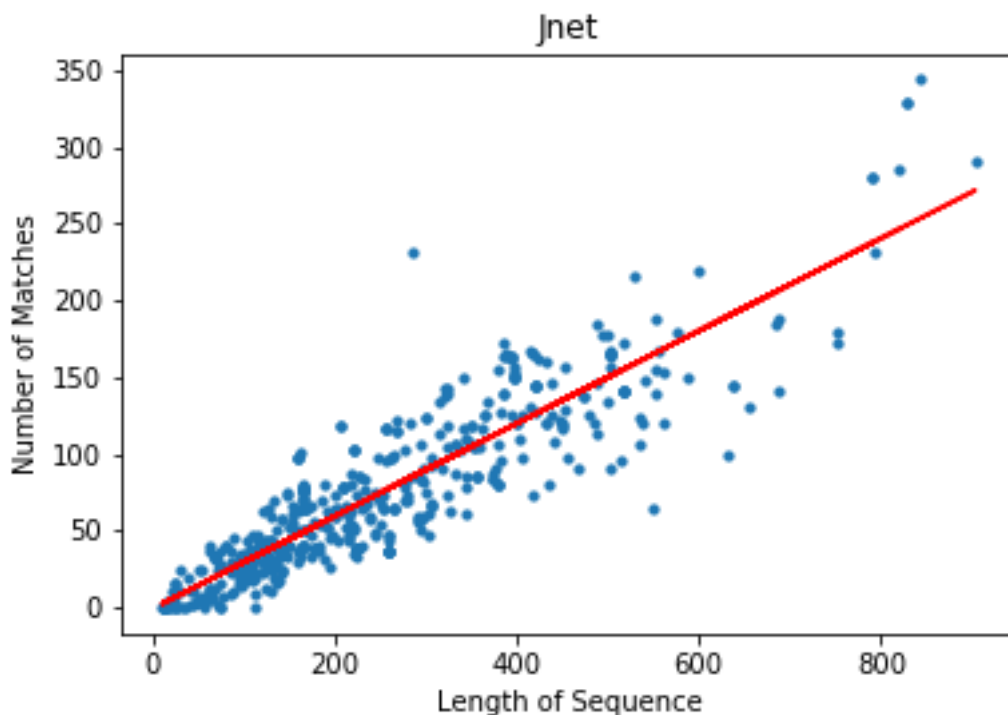
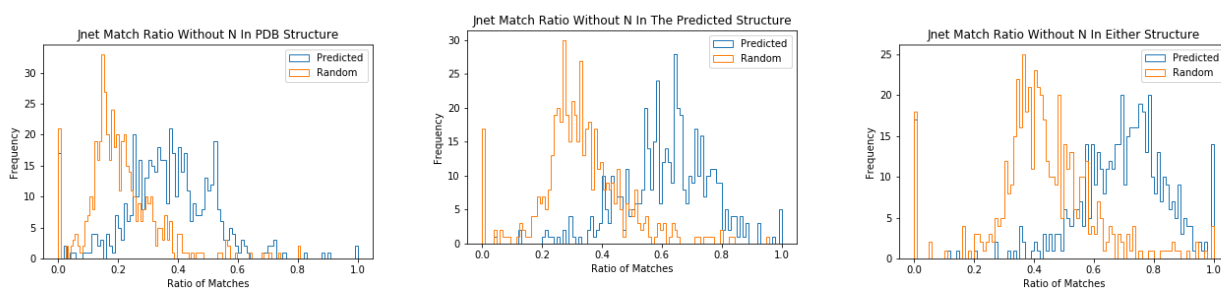


Figure 4.5: The relation between the length of a sequence and the number of correctly assigned structural elements by Jnet.

see a larger separation between the two distributions. This makes sense as Jnet classifies a lot of elements as unknown and we left those out of consideration for this experiment. Figure 4.6c shows a larger shift to the right with a match ratio around 0.8. However, the null distribution has also shifted. Again, this can be explained by the amount of unknown elements in both the target and the prediction structures.

Finally, we notice that there are relatively many sequences that have a match ratio of 0.0. If we take a look at the splatterplot in figure 4.2, we see that they occur in relatively short sequences. The most cases occur if, in the target structure, the number of unknown structural elements is larger than the number of H, E and L combined and if the amount of H, E or L are zero.



(a) Excluding N in the Target structure. (b) Excluding N in the predicted structure.

(c) Excluding all N's.

Figure 4.6: Frequency distributions of the ratios of matching elements with Jnet, using three different metric on how to handle unknown structural elements.

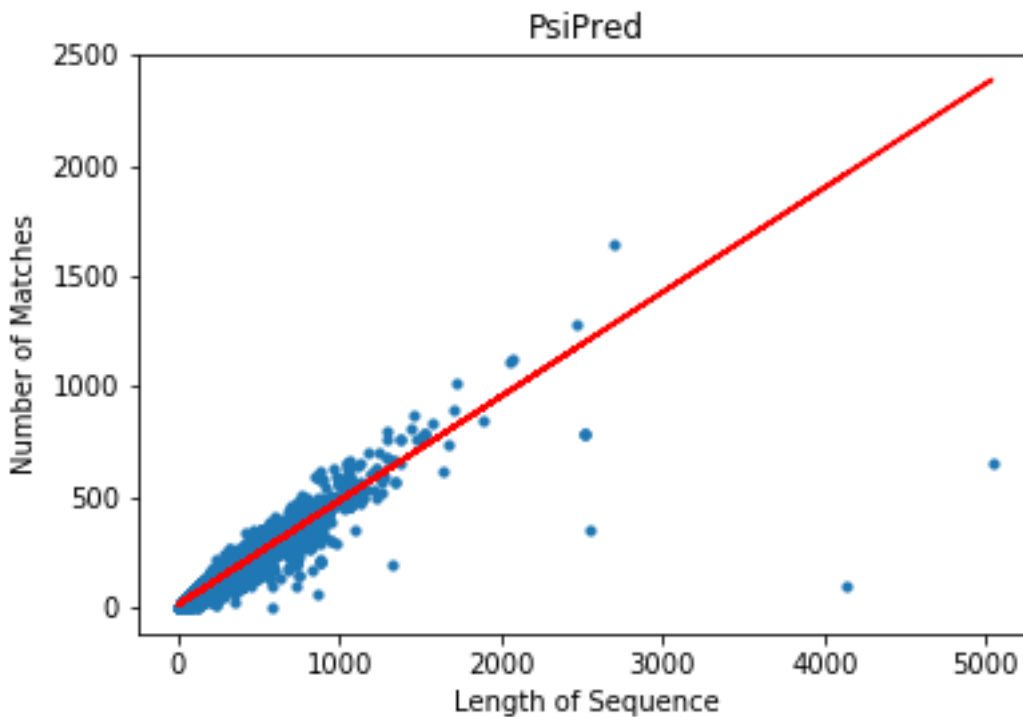


Figure 4.7: The relation between the length of a sequence and the number of correctly assigned structural elements by PsiPred.

4.2.3 PsiPred

Finally we will take a look at the results of the PsiPred algorithm. In figure 4.7 we see that not only does PsiPred handle the most sequences, it also handles longer sequences. Where Agadir fails at sequences longer than 500 and the longest predicted structure for Jnet is 903, the longest structure predicted by PsiPred is over 5000 amino acids long.

In figure 4.8b we see the clearest difference between the null distribution and the predicted distribution. It performs pretty well with the highest concentration between a match ratio of 0.6 and 0.8 and very little cases that have a match ratio under 0.6. There are however a little over 80 cases where the match ratio is 0.0. Most of these cases are smaller sequences ($length \leq 32$) and proportionally a large amount of unknown elements in the target structure. As we indicated earlier, PsiPred does not classify unknown structure elements as such, but classifies them as coil. Which explains why figure reffig:matchPsiPDB looks like it performs worse than the last one. There are no unknown structure elements in the predicted structures, yet we take all the unknown structure elements of the target structure into the equation. It also explains why figure 4.8c looks identical to 4.8b. The match ratio excluding unknown structure elements and the match ratio without unknown structure elements in both structures are essentially the same.

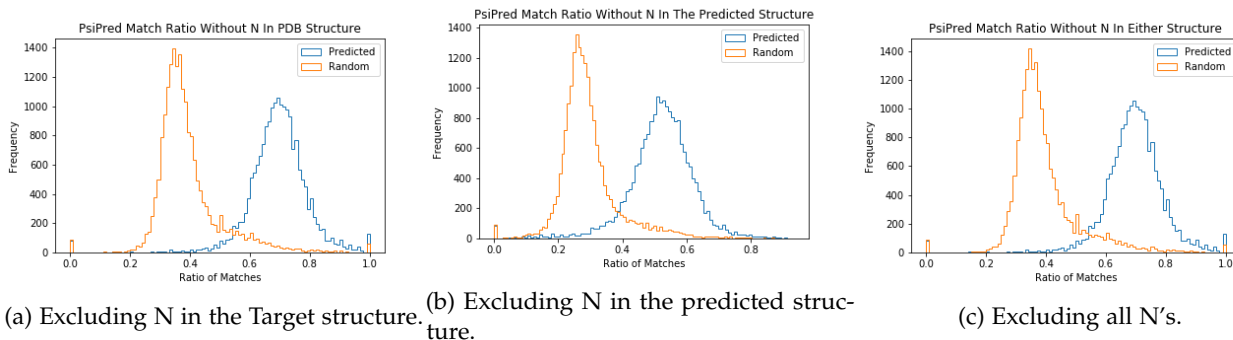


Figure 4.8: Frequency distributions of the ratios of matching elements with PsiPred, using three different metric on how to handle unknown structural elements.

4.3 Summary

The Jaro distance and the Jaro-Winkler distance failed to capture the information content separating the null and predicted distributions. This is due to the small alphabet (H,E and L) and the prevalence of unknown structure elements (N). We have also seen that Agadir is unreliable as a prediction algorithm but still is able to assign, with some accuracy, helical structure elements. Which is what the developers promised in the first place [24]. Jnet performs pretty well but it has trouble with the unknown elements. But because the structures in the PDB have a lot of unknown structural elements, it becomes difficult to train a highly performing algorithm. Finally, PsiPred outperforms the other algorithms. It also predicts structures for longer sequences.

Chapter 5

Discussion And Conclusions

5.1 Discussion

We began this research with the problem that we wanted to test and compare a selection of prediction algorithms against the gold standard of protein structures: the PDB. Although these comparisons are in principle straight forward, there were a number of technical problems that needed to be overcome.

5.1.1 Incomplete Structure Data

We ran 69,058 PDB sequences through four prediction algorithms. However, only 16,937 sequences had a predicted structure (around 25%) and only 88 sequences had a structure predicted by all three algorithms. An interesting question for the future would be: 'why did so many sequences fail to run on the algorithms and is there a reason why there is so little overlap?'. One potential reason why so many sequences failed to produce a structure could be that the batch code used to run all sequence through the prediction algorithms failed at certain moments. One could investigate the batch code but that is beyond the scope of this project. It would be, however, interesting to have structures for all 69,058 PDB sequences.

5.1.2 Structure Similarity

At first we thought that the edit distance would be a useful method to compare the structures. After applying the Jaro and Jaro-Winkler distance algorithm to our dataset we concluded that the results of the prediction algorithm were similar to that of the null distribution. Instead, we developed our own metric by comparing the structural elements in each position with each other to see if they were a match. We used three variations of this metric to take the undetermined structural elements into account.

5.1.3 Validation And Comparisons

We saw that Agadir performed well when we filtered out the unknown structural elements. But it is having troubles if the target structure had a smaller number of helical elements. Unfortunately, this is something you would not know beforehand when predicting the structure of a sequence if there is no target structure to compare the prediction with. Both Jnet and PsiPred perform pretty with a mode of around 75% correct predictions when all undetermined structural elements are left out. All three algorithms have the best performances if we filter out the unknown structural elements in both structures in our metric. The question is: is it fair to compare these structures this way, leaving out a part of the equation? First of all, predicting something as being unknown is not really a prediction. It is better for a algorithm to admit that it does not know a certain position than to predict it wrong. Second of all: even in clinical studies it is hard to determine some structures as is shown by the amount of unknown structural elements in the PDB. So how can the prediction algorithms improve upon this if clinical methods are not 100% sure?

5.2 Conclusion

This research thesis started with the question: In general, how can the performance of protein structure prediction algorithms be systematically validated and compared? To achieve this we created a workflow by comparing a target structure and a predicted structure with our own match per position metric. This method is comprehensive (using all known experimentally determined structure data) and universal (agnostic to the machine learning technology), and can be applied to any existing and potential secondary structure prediction algorithm.

Bibliography

- [1] IBM. Blue gene, the supercomputer designed for grand challenges. http://www-07.ibm.com/solutions/au/downloads/bluegene_03.pdf. [Online; accessed 05-October-2017].
- [2] Gustavo Caetano-Anollés, Minglei Wang, Derek Caetano-Anollés, and Jay E Mittenthal. The origin, evolution and structure of the protein world. *Biochemical Journal*, 417(3):621–637, 2009.
- [3] Frances C. Bernstein, Thomas F. Koetzle, Grahame J.B. Williams, Edgar F. Meyer, Michael D. Brice, John R. Rodgers, Olga Kennard, Takehiko Shimanouchi, and Mitsuo Tasumi. The protein data bank: A computer-based archival file for macromolecular structures. *Archives of Biochemistry and Biophysics*, 185(2):584–591, jan 1978.
- [4] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235, 2000.
- [5] Alexey Drozdetskiy, Christian Cole, James Procter, and Geoffrey J. Barton. Jpred4: a protein secondary structure prediction server. *Nucleic Acids Research*, 43(W1):W389–W394, jul 2015.
- [6] Ambrish Roy, Alper Kucukural, and Yang Zhang. I-tasser: a unified platform for automated protein structure and function prediction. *Nature Protocols*, 5(4):725–738, apr 2010.
- [7] James A Cuff and Geoffrey J Barton. Application of multiple sequence alignment profiles to improve protein secondary structure prediction. *Proteins: Structure, Function, and Bioinformatics*, 40(3):502–511, 2000.
- [8] Emmanuel Lacroix, Ana Rosa Viguera, and Luis Serrano. Elucidating the folding problem of α -helices: local motifs, long-range electrostatics, ionic-strength dependence and prediction of nmr parameters. *Journal of molecular biology*, 284(1):173–191, 1998.
- [9] David T Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of molecular biology*, 292(2):195–202, 1999.
- [10] Renxiang Yan, Dong Xu, Jianyi Yang, Sara Walker, and Yang Zhang. A comparative assessment and analysis of 20 representative sequence alignment methods for protein structure prediction. *Scientific reports*, 3, 2013.

- [11] MongoDB 3.4: Your database evolved. <https://www.mongodb.com/mongodb-3.4>. [Online; accessed 18-September-2017].
- [12] Neal Leavitt. Will nosql databases live up to their promise? *Computer*, 43(2), 2010.
- [13] Python 3.5.4 documentation. <https://docs.python.org/3.5/>. [Online; accessed 16-August-2017].
- [14] Python driver (pymongo), getting started with mongodb 3.0.4. <https://docs.mongodb.com/getting-started/python/client/>. [Online; accessed 16-August-2017].
- [15] James Turk and Michael Stephens. jellyfish 0.5.6 : Python package index. <https://pypi.python.org/pypi/jellyfish>. [Online; accessed 31-August-2017].
- [16] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation*, volume 3, pages 73–78, 2003.
- [17] Fred J. Damerau. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176, March 1964.
- [18] Eric Brill and Robert C. Moore. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, ACL '00*, pages 286–293, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.
- [19] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string metrics for matching names and records. 10 2003.
- [20] Pandas. Pandas about. <https://pandas.pydata.org/about.html>. [Online; accessed 18-September-2017].
- [21] Stephen F. Altschul, Thomas L. Madden, Alejandro A. Schffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.
- [22] Ucl department of computer science. <http://bioinf.cs.ucl.ac.uk/index.php?id=779>. [Online; accessed 22-September-2017].
- [23] Jnet: A neural network protein secondary structure prediction method. <http://www.compbio.dundee.ac.uk/jpred/legacy/jnet/how.html>. [Online; accessed 20-September-2017].
- [24] An algorithm to predict the helical content of peptides. <http://agadir.crg.es/>. [Online; accessed 22-September-2017].