UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduado en Ingeniería de la salud

Aplicación de técnicas de Deep Learning al Alineamiento Múltiple de Secuencias

Deep Learning and its applications in Multiple Sequence Alignment

Realizado por

Fiorella Piriz Sapio

Tutorizado por

Antonio Jesús Nebro Urbaneja
José Manuel García Nieto

Departamento

Lenguajes y Ciencias de la Computación

MÁLAGA, 25 DE JUNIO DE 2021

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADUADO EN INGENIERÍA DE LA SALUD

Aplicación de técnicas de Deep Learning al Alineamiento Múltiple de Secuencias

Deep Learning and its applications in Multiple Sequence Alignment

Realizado por
Fiorella Piriz Sapio

Tutorizado por
Antonio Jesús Nebro Urbaneja
José Manuel García Nieto

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, 25 DE JUNIO DE 2021

Fecha defensa: julio de 2021

Abstract

In the last decade, Deep Learning (DL) has been one of the most successful approaches in various fields of study, including Bioinformatics. However, there are not many state-of-art DL proposals in Multiple Sequence Alignment (MSA), an essential tool in the day-to-day life of biologists. In particular, MSA is one of the most challenging problems in Bioinformatics, as it allows to discover the similarity between multiple biological sequences (DNA, RNA or proteins). Moreover, it has many applications in comparative structural and functional analysis. However, the alignment may differ when using one or another application, so the scoring may not be optimal, which is the goal of MSA. Therefore, it is a difficult task to choose which alignment to perform depending on the protein family.

Therefore, this thesis proposes a tool that compares different Machine Learning models, such as Decision Tree, Random Forest or Gradient Boosting, with two Convolutional Neural Network (CNN) architectures. Its objective is to demonstrate that Deep Learning is an efficient procedure to classify protein families based on their average pairwise identity percentage (PID). These decision making models determine which of the four selected Multiple Sequence Alignment (MSA) tools: MAFFT, Muscle, ClustalW2 and T-coffee, will provide the most accurate output alignment.

Finally, to display the result of this project, we have implemented a Web application called MachineLAlign (open code avilable here) that allows to compute the alignment of multiple sequences with any of the trained and tested models.

Keywords: Multiple Sequence Alignment, Deep Learning, Machine Learning, Performance analysis

Resumen

En la última década, Deep Learning (DL) ha sido una de las técnicas más empleadas en diversos campos de estudio, incluida la Bioinformática. Sin embargo, a día de hoy no existen demasiados estudios en los que se aplique DL en para el Alineamiento de Secuencias Múltiples (MSA), una herramienta esencial en el día a día de disciplinas como la Biología y la Bioinformátic. En concreto, los problemas de MSA son muy relevantes en Bioinformática ya que permite descubrir similitudes entre múltiples secuencias biológicas (ADN, ARN o proteína). Además, tiene muchas aplicaciones en la comparación de estructuras y el análisis funcional. Sin embargo, la alineación resultante puede diferir de usar una u otra aplicación, por lo que, el resultado puede no ser óptimo, lo cual es el objetivo de MSA. Por tanto, es una tarea difícil elegir qué alineación realizar dependiendo de la familia de proteínas.

Por ello, en este trabajo de fin de grado proponemos una herramienta para comparar diferentes modelos de Machine Learning, como Decision Tree, Random Forest o Gradient Boosting, con dos arquitecturas de redes neuronales convolucionales (CNN). El objetivo es demostrar que el aprendizaje profundo es un procedimiento eficaz para clasificar familias de proteínas en función del promedio de identidad porcentual por pares de secuencias(PID).Estos modelos de toma de decisiones determinan cuál de las cuatro herramientas seleccionadas para la Alineación de Secuencias Múltiples (MSA): MAFFT, Muscle, ClustalW2 y T-coffee, proporcionará la alineación final más precisa.

Finalmente, para representar el proyecto, hemos implementado una aplicación Web llamada MachineLAlign (código abierto disponible aquí) que permite obtener el MSA con cualquiera de los modelos entrenados y evaluados.

Palabras Clave: Alineamiento Múltiple de Secuencias, Deep Learning, Machine Learning, Evaluación de rendimiento

Contents

List of Figures

List of Tables

Acronyms

Acronym	Description
AI	Artificial Intelligence: Area of computer science that aims to create machines that performs tasks simulating humans' behaviour [1].
BiLSTM	Bidirectional Long Short-term Memory
CNN	Convolutional Neural Network.
CSS	Cascading Style Sheets
DL	Deep Learning: Data science field inspired by the structure of the human brain, where neurons transmit signals from nerve impulses.
DNA	Deoxyribonucleic Acid: Molecular structure that contains the genetic information.
DNN	Deep Neural Network
HTML	HyperText Markup Language
KNN	K Nearest Neighbours
HMM	Hidden Markov Model: Generative models utilized for creating probabilistic matrix, which has as input labeled linear sequences[2].
ML	Machine Learning: Branch of AI that has the objective of building tools that learn automatically from input data.
MSA	Multiple Sequence Alignment
NP	Non-deterministic Polynomial
PID	Percentage Identity: Measurement of similarity between biological sequences.
RL	Reinforcement Learning
RNA	Ribonucleic Acid: Nucleic acid that take part in protein synthesis.
RF	Random Forest

Table 1: Acronyms used along the document and their description.

1 Introduction

Multiple Sequence Alignment (MSA) is one of the most important applications in Bioinformatics for the reason that it has many applications in the field of computational biology, such as protein structure prediction, conserved patterns identification, phylogenetic analysis, and more [3].

It consists in aligning a large number (at least three) of DNA, RNA or protein sequences, in order to identify the regions they shared. These findings might reveal functional, structural or evolutionary relationships, which provide more biological information than pair-wise alignments (more details in Annex 1).

1.1 Motivation

Although there are many approaches to MSA, it is a problem of NP (non-deterministic polynomial time) computational complexity and current tools tend to provide poor results, especially when the problems are composed of many long sequences. That is the reason why the development of an implementation that optimizes the alignment of multiple sequences could be very helpful and powerful, especially for researchers and scientists facing with MSA problems on a day-to-day basis.

Nevertheless, there are many ways to develop this tool, but when we considered this problem, Deep Learning (DL) techniques came to mind. The recent success of DL in fields such as image recognition and natural language processing, provides the opportunity to apply this method in other areas of study just as Bioinformatics. Thus, our approach seeks to enhance the performance of MSA implementation, leading to a better alignment of the biological sequences.

1.2 Objectives

The objectives of this Final Degree Project could be represented in four points:

- Analyze the existent projects related to the application of Deep Learning to MSA. In detail, I have chosen DLPAlign as the reference study to review in depth [4].
- Replicate the model implemented in the previous source.
- Test the reliability and robustness of Deep Learning implementation to align multiple sequences. For this purpose, other ML approaches will be applied to compare the outputs.
- Develop a tool for Multiple Sequence Alignment using the previously trained models. This application should include the training and validation of the different models generated, the classification of biological sequences, and the alignment of these sequences according to the classification obtained.

Since MSA is a tool (in Biology or Bioinformatics) that has been used for many years and there are various applications that perform this type of alignment, what is sought with this project is to propose an innovative and improved technique that optimises the results of the alignments. To do so, we will investigate which is the best approach for this type of problem, as well as which are the parameters that will allow us to obtain greater effectiveness with our data set. Therefore, once the tool is developed, we will try to evaluate whether our goals are achieved or not.

1.3 Technologies used

Python is the modern programming language and it is widely used in Bioinformatics, thus we will employ some of the software libraries available to apply AI in Python 3.8.5.

In order to develop ML models we have utilized *sklearn*, a module that integrates a wide range of Machine Learning algorithms to solve supervised or unsupervised problems. "This package focuses on bringing machine learning to non-specialists using a general-purpose high-level language", as expressed in *Scikit-learn: Machine Learning in Python* [5]. This pack-

age is very useful because it allows to train different models and tuning their parameters implementing almost the same functions, so the flow can be automated.

In the other hand, we have used Keras on version 2.4.3 as the high-level API of Tensorflow 2.4.1 to generate Deep Learning models as they provides intuitive and visual tools for building and validating network architectures.

Finally, to develop the Web application we have used libraries such as Flask to create the API Res, HTML to create the visualisation, CSS to control the appearance and JavaScript to add functionalities. In addition, for the MSA representation, we have make use of pyMSA [6], a software application to score MSA developed by a University of Malaga student.

All the above indicated technologies are detailed at Table 2.

1.4 Methodology

In this section we describe the steps followed to implement this project as well as the tools used to make it possible.

As the goal of this work is to prove that the application of Deep Learning in Multiple Sequence Alignment is reasonable, we first conducted an comprehensive study of the current "sate of art" related with this type of models. After choosing DLPAlign as the main reference project, we tried to understand the fundamentals of this project in order to replicate it.

Thus, our method tests the accuracy of DL architecture applied to the MSA by comparing results with other Machine Learning models such as K-Nearest Neighbours, Random Forest, Decision Tree, AdaBoost or Gradient Boosting. Our hypothesis is that DL is a powerful tool that performs perfectly in many areas of study, but we want to ensure that it can be better that existing ones in this case.

We part from a large dataset composed by 954,854 rows which give information about the length of the sequence, the chain of amino acids that form it, the PID or percentage identity of each protein family, the Sum of Pairs (SoP) and the classification of the sequence. It is worth noting that the Class value can be between 0 and 3 as we proposed 4 different tools to classify sequences. In addition, it is important to remark that the data has been augmented by coupling pairs of sequences from the same protein family.

Once had prepared the data, we started to implement the models, passing through three phases (Figure 1):

Technology name	Usage description
	Main packages used in Python to apply AI.
Scikit-learn or Sklearn	Utilized for creating, training and testing ML models
Tensorflow	”End-to-end open source machine learning platform” [7].
Keras	Python library for Deep Learning
	Instruments employed for Web application development.
Flask	Framework written in Python that allows to easily create Web applications
HTML	Programming language that is essential in web pages.
CSS	Programming language that allows to add styles to the web page content.
JavaScript	Programming language used to add functionality to the Web.
Boostrap	”The most popular CSS Framework for developing responsive and mobile-first websites” [8].
	Instruments employed for Aligning Multiple Sequences.
ClustalW, MAFFT, T-Coffee, Muscle	MSA tools.
pyMSA	Software employed for scoring alignments.
	Other.
Linux	Operative System.
Visual Studio Code	Code editor used for creating the project.
GitHub	Projects repository where MachineLAlign is stored.
Overleaf	Cloud text editor for LaTeX.
Scopus	Database of bibliographic references and citations.
Mendeley	Software that allows to manage references and citations.

Table 2: Table with main technologies used to carry out this project.

- Hyperparamenter tuning: Choose a set of parameter for a learning algorithm, having the best performance of the model (optimum).
- Cross Validation: Procedure applied to ML models that provides information about the generalisation of a classifier.
- Model Comparison: It allows to choose the model which has an optimum performance for the given data.

Then, having trained and validated the models, we can compare the scores, so as to decide which of them is having the most accurate performance.

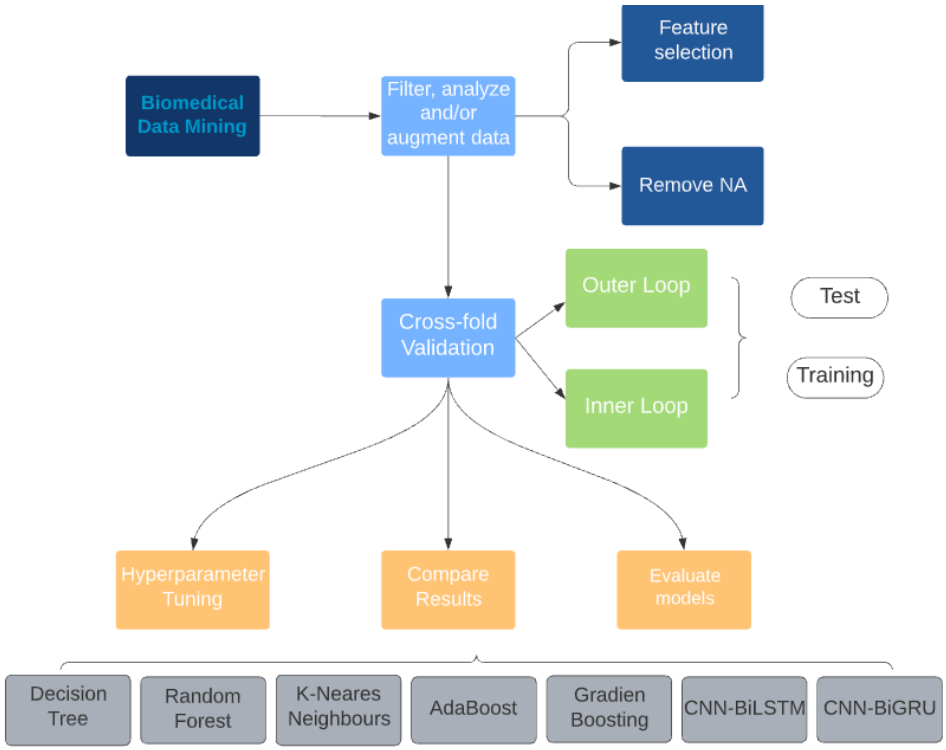

Figure 1: Description of the phases followed throughout the project

In addition, we decided that the best way to display the results of this project is to develop a Web application where users have the opportunity to easily align sequences with the ML architectures generated.

1.5 Structure of the document

This work has been partitioned in seven parts which include:

- Introduction to the project which contextualise the topic, and describes what are the aims to reach. Furthermore, it includes the Methodology, the part of the memory where methods, techniques and procedures are expressed so as to communicate readers what has been done.

- State of Art. It includes all the information about the current state of Multiple Sequence Alignment approaches and Artificial Intelligence applied to this field. Additionally, we describe the MSA tools used in this study.

  Implementation. It is used to express the development process and to show some code

snippets to make the document easier to understand.

- Results and Discussion. The outputs obtained thorough the development of the study are exposed in this part to show key findings. In addition, we discuss the results and compare with other studies.

- Finally, the Conclusion holds the overview of the project.

We have added a section for Annexes to add supplementary information that may deliver relevant information about MSA and DL.

2

State of Art

In the last decades, MSA has been proven to be a powerful tool to discover relationships between sequences, so it has many applications in the area of molecular biology. However, it still has some drawbacks, so many efforts have been made in the field of Bioinformatics research to develop new approaches to optimise its performance.

2.1 Multiple Sequence Alignment technologies

Historically the most widely used applications for Multiple Sequence Alignment have been **ClustalW**[9], a program for progressive alignment based on Feng - Doolittle algorithm, Muscle, which calculates an iterative alignment, as well as **MAFFT** [10], one of the applications that produces the best scores. **T-coffee** [11] is also based on progressive algorithms, but unlike ClustalW it aims to solve the errors produced in the pairwise alignment.

However, the most advanced tools to date are based on Hidden Markov Model (HMM), which offers a variety of refinements in the guide tree construction. **Clustal Omega** [12] is an example of this algorithm that uses HMM profile-profile to calculate posterior the posterior probability matrix. Here below we will describe the MSA instruments employed to align sequences.

2.1.1 T-Coffee

Tree-based Consistency Objective Function for alignment Evaluation (T-coffee) consists of a method for aligning multiple biological sequences based on progressive alignment. It was published in 2000 to provide a simple and flexible tool using different data sources [11] .

First of all, data from a library of local and global pairwise alignments are provided. Next, the input dataset is employed to guide the progressive alignment process, “where intermediate alignments are then based not only on the sequences to be aligned next but also on how all of

the sequences align with each other" [13]. This is a feature of the progressive algorithm called Fend-doolittle [14]. Intermediate alignments are then based not only on the sequences to be aligned next, but also on how all of the sequences align with each other (see Figure 2).

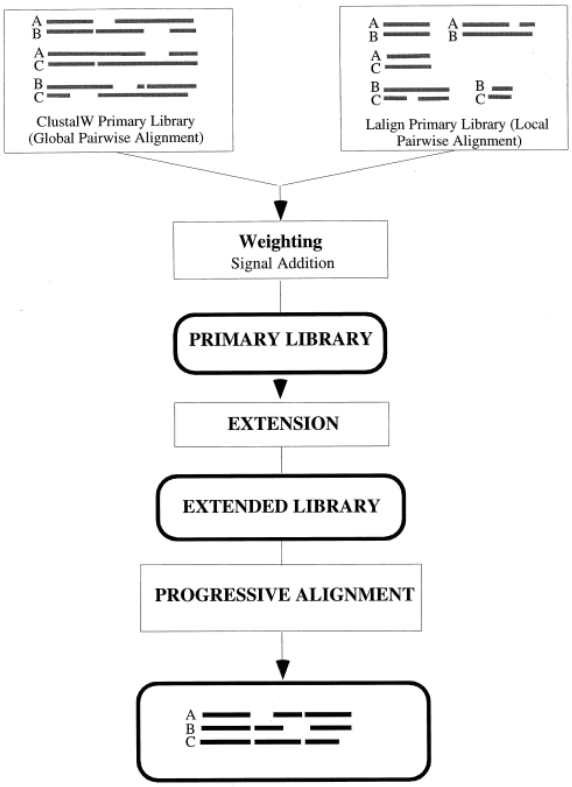

Figure 2: T-Coffee multiple alignment procedure. Taken from T-Coffee: a novel method for fast and accurate multiple sequence alignment.[13]

Thus, T-Coffee, has the ability to take information from all the sequences, allowing it to be fast and robust.

2.1.2 MAFFT

MAFFT [10] or Multiple Alignment using fast Fourier transform, is an approach developed in 2002 that drastically reduces the CPU time. It is based on an innovative technique that identifies homologous regions by the Fast Fourier Transforms (FFT) and combines progressive and iterative refinement heuristics.

The authors believe that the frequency of amino acid substitutions depends on the physico-chemical properties, so they start by calculating the correlation between two amino acid se-

quences to then find homologous regions between the sequences by FFT analysis.

Moreover, this tool uses a scoring metric based on a normalized similarity matrix, which together with a simpler gap penalty technique, allows similar results to be obtained while consuming less CPU time.

2.1.3 CLUSTALW

It is an approach to MSA introduced in 1994 that implements the progressive alignment method. It follows four steps [15]:

- Assign individual weights to each sequence so as to decrease the weight among similar sequences and increase it among the divergent ones.
- Calculate a distance matrix according to the divergence of the sequences.
- Compute a guide tree based on distance matrix.
- Progressive alignment of the sequences in line with a guide tree.

In summary, CLUSTALW [16] is a simple but appropriate device that have been improved over time, being CLUSTALW2 the current version.

2.1.4 MUSCLE

This is another MSA instrument that we have used in our classification problem. It was created in 2004 with improved speed and accuracy compared to previously evoked tools. It is built on "fast distance estimation using kmer counting, progressive alignment using a new profile function we call the log-expectation score, and refinement using tree-dependent restricted partitioning", as expressed in *MUSCLE: multiple sequence alignment with high accuracy and high throughput* [17].

As mentioned before, MUSCLE proposes a new strategy for distance matrix calculation based on kmer distance and Kimura distance, which are clustered using UPGMA.

2.2 Reinforcement Learning approaches

Besides, other studies are based on Reinforcement Learning (RL), an area of Machine Learning that "combines artificial neural networks with an architecture that enables software-defined

agents to learn the best actions possible in virtual environment in order to attain their goals", as expressed in *A Beginner's Guide to Deep Reinforcement Learning* [18]. It is the case of **RLALIGN** [19], a Bioinformatic tool that aims to align multiple sequences using RL method, which obtained powerful outcomes for complex combinatorial problems. Furthermore, this approach, unlike the traditional ones, is independent of the scoring metric, so it can be easily used across the board.

However, it is not the only implementation of RL towards MSA; *Mircea et al.* [20] also proposed a tool focused on DNA sequences that managed to "preserve the incremental pairwise nature of the major currently employed MSA tools (such as ClustalW, MAFFT) while exploring the search space in an effective reward-targeted manner".

2.3 Deep Learning approaches

Otherwise, Deep Learning models have been used in several biomedical scenarios such as Ray-X image classification, pattern recognition, electrocardiogram (ECG) signal processing, etc. and last year it was applied to MSA by **DLPAlign** [21], a study which employed a DL model to classify protein families to be aligned using a posterior probability matrix calculation or another, determined by the predicted output. The latter consists of the first step of progressive alignment, so depending on the algorithm used, the result is different and should be optimal. The authors tested its performance by comparing the scores for three empirical benchmarks with eleven popular MSA tools and DLPAlign always produced an improvement. This project will be explained in depth later.

Another example is **rawMSA** [22], a end-to-end deep learning approach for protein structure prediction in which the input to the Deep Neural Network is the whole MSA. The results of this study reveal that it could be an optimised implementation for representing evolutionary information.

After researching among a list of eighth publications related to Artificial Intelligence and MSA, we decided that the best reference to start this work was DLPAlign. Thus we will give an introduction to this project and its conclusions.

2.3.1 DLPAlign

As discussed above, *DLPAlign: A Deep Learning based Progressive Alignment for Multiple Protein Sequences* is a paper published on July 2020 by Mengmeng Kuang to propose an innovative tool to improve the accuracy of the MSA. It is built on a decision-making model formed by a Convolutional Neural Network and a Bidirectional Long Short-term memory, whose output establishes the progressive alignment method.

The author considers that for a specific protein family, the accuracy of the alignment tools employed can be slightly different, especially for those families with low similarity. Thus he presents an approach that tries to optimise the parts of the progressive algorithm that could provide the best improvement.

A progressive alignment is usually composed by the following parts:

1. Posterior probability matrix.
2. Distance matrix calculation.
3. Guide Tree generation.
4. Consistency transformation.
5. Refinement.

After testing different algorithms belonging to ProbCons [23], Probalign, MSAProbs, GLProbs and PnpProbs, for parts 1, 3 and 4, the author concluded that the section to optimise was the first one since the final result was determined by the posterior probability matrix.

After training with 6 different architectures: CNN, CNN + RNN, CNN + LSTM, CNN + BiLSTM, CNN + GRU and CNN + BiGRU; a decision-making model based on CNN + BiLSTM was created, as it was the one that achieved a highest macro average f-score ($Fmacro$).

Therefore, this model was integrated into existing progressive alignment methods to align sequences of a family F based on the computed posterior probability matrix. The process followed can be observed in Figure 3 .

Finally, to determine the accuracy of DLPAlign it was compared with eleven tools: QuickProbs, PnpProbs, GLProbs, MSAProbs, Probalign, ProbCons, PicXAA, MAFFT, MUCSLE, Clustal

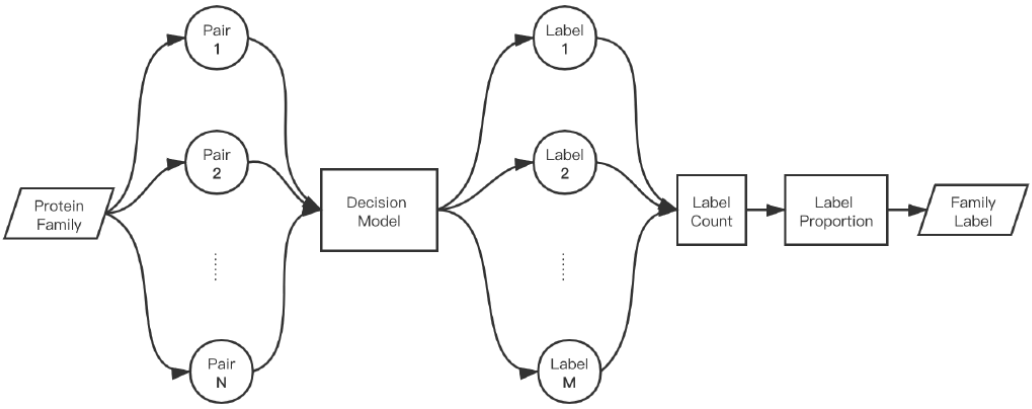

Figure 3: Process followed to determine the family label for aligning the sequences of a protein family *F*. [21]

Omega and T-Coffee; sequences were taken from three empirical benchmarks BAliBASE 3.0, OXBench 1.3 and SABmark 1.65 and the metric used was Total-Column score.

As concluded in the paper, "results show that DLPAlign can get the best total-column scores on the three benchmarks. When evaluated against the 711 low similarity families with average PID 30%, DLPAlign improved about 2.8% over the second-best MSA software" (Figure 4) .

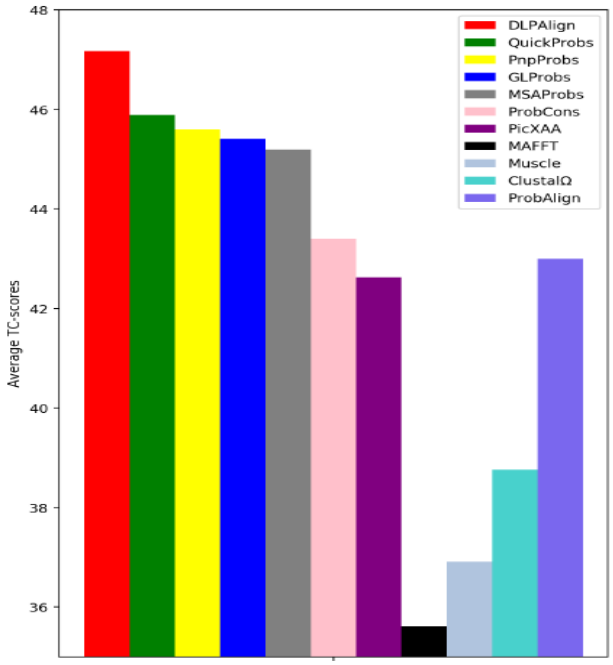

Figure 4: TC-score over 711 families with PID $\leq$ 30%. [21]

3 Implementation

In this chapter we will describe the development process carried out to test the effectiveness of our application MachineLAlign. We started by stratifying the data to reduce the size but keeping the stratification, then we created the models, tuned their parameters, and trained and validated them. Finally we compared all the results and designed and developed the Web application.

3.1 Data stratification

Due to the large size of the input data, and in order to refine the hyperparameters of our models, we decided to divide data into smaller strata to make it computationally feasible to run the algorithms. Moreover, We tried to maintain the same proportion of classes (observed at Figure 5) as in the original population by applying stratified sampling.

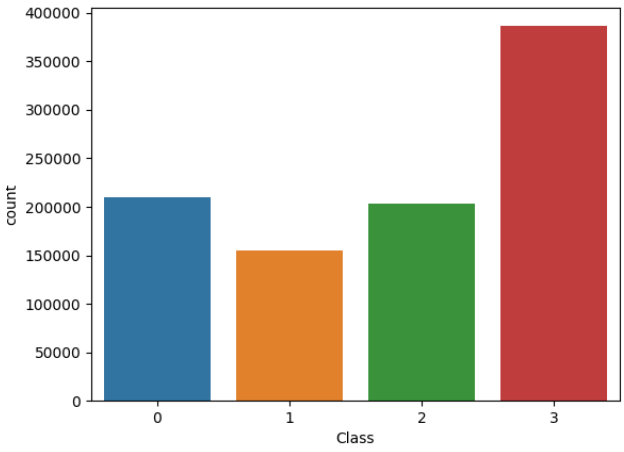

Figure 5: Distribution of classes in the original dataset to be classified.

In the first place, we upload and visualize the input data to finally stratify data:

```
stratified_split = StratifiedShuffleSplit(n_splits = 2, test_size = 'size'
    df = pd.read_csv('../resources/full_pair_data.csv')
print(df.shape)
print(df.head())
df.info()
```

```
(954854, 8)
No.   Lengths        Veterbi   ...      Emit      SoP      Class
0     1       193  BBBBBBBBBBB  ...   ...  0.004267  0.072539      3
1     2       265  YBBBBBBBBBB  ...   ...  0.004601  0.505660      3
2     3       199  YYYYYBBBBBB  ...   ...  0.004019 -0.020101      3
3     4       227  YYYYYYYYBBB  ...   ...  0.003635  0.096916      3
4     5       260  YYBBBBBBYYY  ...   ...  0.003096  0.292308      3
```

```
stratified_split = StratifiedShuffleSplit(n_splits = 2, test_size = 'size'
    , random_state = 64)
```

Therefore, we trained input data with three different sizes, 0.01 that represents 9,000 rows, 0.05 implies around 45,000 rows and 0.1 that generated an input of 90,000 entries. However, we did not keep increasing its size because the results were directly proportional to data augmentation.

3.2 Hyperparameter tuning

As it already mentioned, this technique allows to improve the performance of a model by choosing a set of parameters that minimises the loss function on given data.

To achieve that, we followed the next steps:

1. Firstly, it is necessary to choose the parameters to be tune and which values to give them.

2. Perform a grid search to generate candidates from the previously specified grid of parameters. All possible solution will be trained, so the number of parameters and values to

be trained has to be reasonable. To achieve this goal we used GridSearchCV that gives the opportunity to implement Cross Validation.

3. Once all possibilities are trained, we select the one that performs best.

4. Predict output with the best estimator in order to compute the accuracy of the model prediction.

5. Finally, we decided to plot the confusion matrix to see how well classes are classified.

This procedure has been implemented with the following code:

```
# Apply grid search with 10 CV
model_grid_search = GridSearchCV(estimator = model, param_grid = prameters_
    grid, cv = 10, n_jobs = -1)
# Train models with given parameters
model_grid_search.fit(X_train, y_train)
# Select the best estimator
best_model = model_grid_search.best_estimator_
# Predict output
prediction = best_model.predict(X_test)
# Compute accuracy
accuracy_value= accuracy_score(y_test, prediction)
# Plot confusion matrix
confusion_matrix_model = confusion_matrix(y_test, prediction)
```

As it was stated in step 1, we have to define the parameter grid, which will be different for each model. This is a complicated task, as there is no stipulated direction for choosing parameters or values; consequently, we have to investigate which are the most common features and values adjusted for each specific model and modify those that do not improve the performance of the given model.

The grid used for each model is shown below.

1. Decision Tree: It is a supervised learning method that employs tree-like model of decisions in order to classify data. Having this in mind, it was decided to find optimum values for the maximum depth of the tree (max_depth), the minimum number of samples to split an internal node (min_samples_split) and the criterion to measure the quality of a split [24].

```
parameter_grid_decision_tree = {
    "max_depth": range(1 ,10, 2),
    "min_samples_split": range(2, 10, 2),
    "criterion": ["gini", "entropy"]
}
```

2. Random forest: It consists of a classification algorithm composed by a large number of decision trees acting together. Each tree makes a prediction and the best class becomes the model prediction. Therefore, the parameters are very similar to the previous ones, minus 'n_estimators', that refers to the number of trees in the forest.

```
parameter_grid_random_forest = {
    "criterion": ['gini', 'entropy'],
    "n_estimators": [200, 300, 400],
    "max_depth": range(20, 50 ,10),
}
```

3. Gradient Boosting: As it happened in RF, gradient boosting consists of a set of decision trees that are sequentially trained to improve the performance of the previous tree. In other words, it is an ensemble model because it combines several algorithms.

   In this case, 'learning_rate' is one of the most important parameters, as it allows to determine how fast a model learns and ,consequently, regulates the risk of overfitting.

```
parameter_grid_gradient_boosting = {
    "n_estimators": range(100, 400, 100),
    "max_depth": range(2, 9, 2),
    "min_samples_split": range(400, 700, 100),
    "learning_rate": [0.1, 0.2]
}
```

4. AdaBoost: Adaptive Boosting is a meta-estimator classifier that combines multiple decision trees with a single split. "It works by putting more weight on difficult to classify

instances and less on those already handled well" [25].

It uses a boosting algorithm in ensemble method just like Gradient Boosting, so the most important hyperparameters are similar to the above mentioned.

```
parameter_grid_adaboost = {
    "n_estimators": range(100, 400, 100),
    "learning_rate": [0.2, 0.3, 0.4]
}
```

5. K Nearest Neighbour: Supervised learning algorithm that is commonly used for classification and regression problems. It tries to find the most similar category for the new data characteristics from those available.

   In the present case, 'leaf_size' is an important parameter to optimise as it affects to the construction speed and the memory required. In addition, it is relevant to determine the number of neighbours (n_neighbours) and the "power parameter for the Minkowski metric" [26].

```
parameter_grid_knn = {
  "leaf_size":range(1, 50, 10),
  "n_neighbors": range(2, 16, 3),
  "p":[1, 2]
}
```

6. CNN BiLSTM and CNN: In the case of DL architectures, we decided to tune the most common parameters for running the models, such as the number of times the entire dataset pass through the NN (epoch), the batch size or the optimizer used. For both models we employed the same values for epochs and batch_size but the optimizer was a bit different.

```
parameter_grid_CNN_BiLSTM = {
  "epochs" : [5, 10, 15],
    "batch_size" : [512, 1024],
    "optimizer" : ['rmsprop', 'nadam']
}
```

```
parameter_grid_CNN = {
  "epochs" : [5, 10, 15],
    "batch_size" : [512, 1024],
    "optimizer" : ['adadelta', 'nadam']
}
```

3.3 Model evaluation and Sequence Alignment

Once we have obtained the best combination of parameters for the final models, we can train and make predictions with the entire dataset (more than 900,000 entries) in order to evaluate their performance. In addition, we will be able to compare models and decide which of them has the best performance. This part will be discussed in the following section (results and discussion).

In the other hand, the Sequence Alignment is the main step of our project since the goal is to apply DL, and in general ML, to MSA. For that reason we designed a program to read fasta sequences from files or text that are aligned with one of the MSA tools mentioned in State of Art. To decide which tool to use, at least one of the previously evaluated models must be selected.

Finally, a file with the resulting alignment and its correspondent scores is automatically downloaded.

3.4 Web Application

We consider the best way to display the work done is by developing a Web application that allows to align sequences with one or more selected ML architectures.

The interface provides a selector to attach files from the user's device, but user can also paste the multiple sequences in the text area. At least one of the models must be selected to provide the alignment, which is carried out when the button 'Send' is pressed. All these items can be observed at Figure 6.

Finally, the alignment result for each selected model is displayed. Moreover, the user has the option the user has the option of downloading a file with the generated output, as repre-

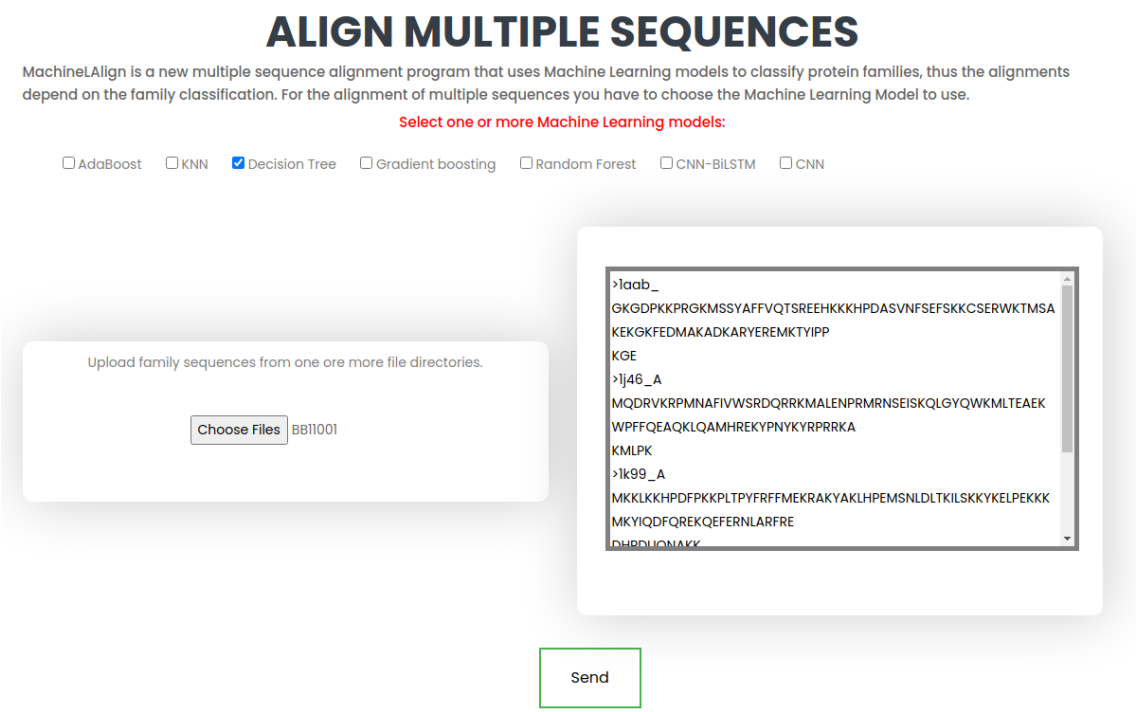

Figure 6: User interface to perform alignments in the developed Web.

sented in Figure 7 .

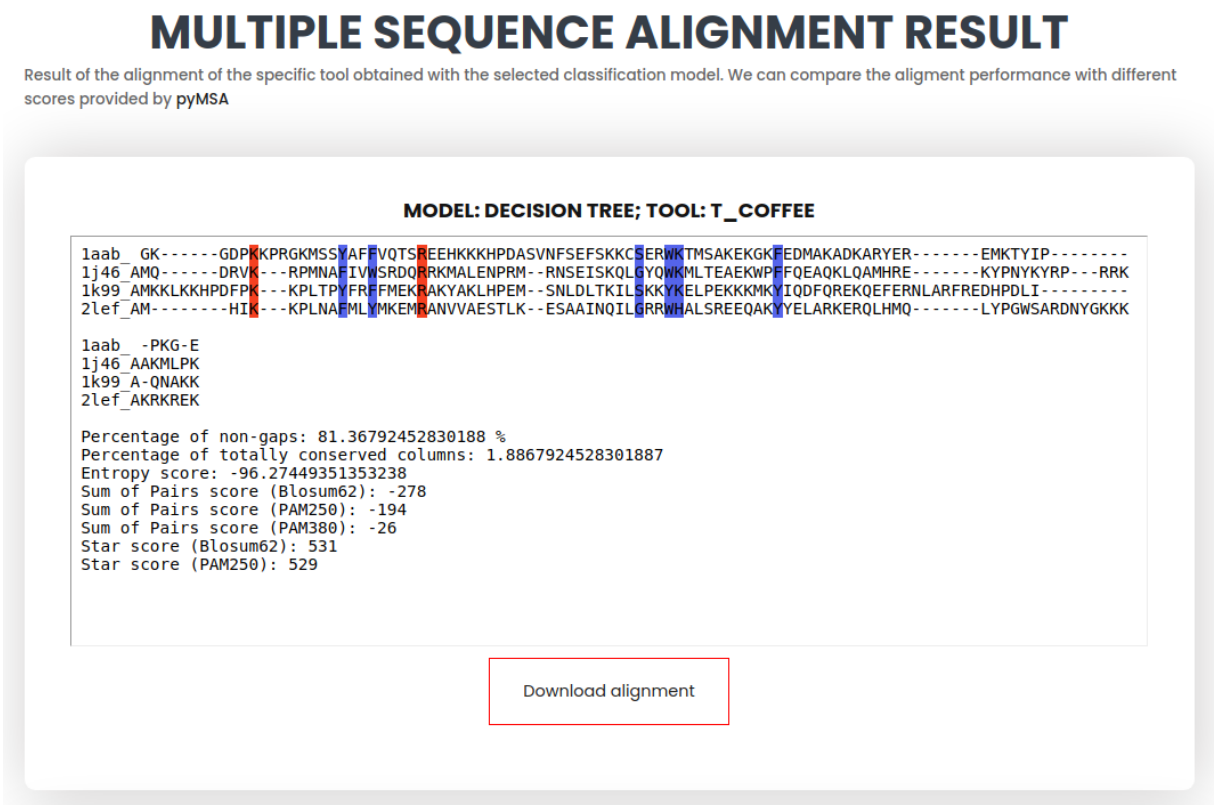

Figure 7: User interface to visualise the result of MSA with each selected model.

4

Results and Discussion

This part of the memory is devoted to explain the quality of the results and to give sense to the development process. We will also compare the results obtained.

As above-mentioned, we have developed different ML models aiming to study whether DL is the best approach to MSA. However, doing so is not so simple if we want to create a robust model that generalizes well. In other words, we need to avoid overfitting, one of the most common problems in AI that occurs when the model is trained better than it predicts.

However, there are some solutions that can be implemented to avoid it such as k-fold cross-validation. This technique consists of splitting data into k-folds, among which one set is kept for training and the another for testing. In addition, the training set is divided into training and test so as to make an honest estimation of the model hyperparameters (hyperparameter tuning).

Having that in mind, we have tried to obtain the best possible results with each model by training and testing them correctly. The accuracy of the models for each given data dimension is represented in Table 3.

The first impression we get from that table is that, in general, the larger the size of the dataset, the better the model predicts the output class. This fact has an explanation, since the model has more opportunities to betrain and, therefore, it performs a more accurate prediction.

In particular, CNN-BiLSTM is the model that has the highest accuracy with the whole dataset (Figure 8); however, with the stratified data, Gradient Boosting is much better. Therefore, it means that the CNN-BiLSTM architecture depends on large datasets.

Gradient Boosting, on the other hand, has an outstanding performance for the decision-making problem, having a progressive increase in accuracy as the input data size increases.

	9,000 entries	45,000 entries	90,000 entries	900,000 entries
Decision Tree	0.636	0.6996	0.701	0.711
Random Forest	0.724	0.794	0.826	0.870
Gradient Boosting	**0.750**	**0.812**	**0.831**	0.864
AdaBoost	0.656	0.664	0.674	0.676
K Nearest Neighbours	0.730	0.802	0.828	-
CNN-BiLSTM	0.232	0.717	0.783	**0.883**
CNN	0.589	0.756	0.796	0.861

Table 3: Accuracy of the models with different data dimensions

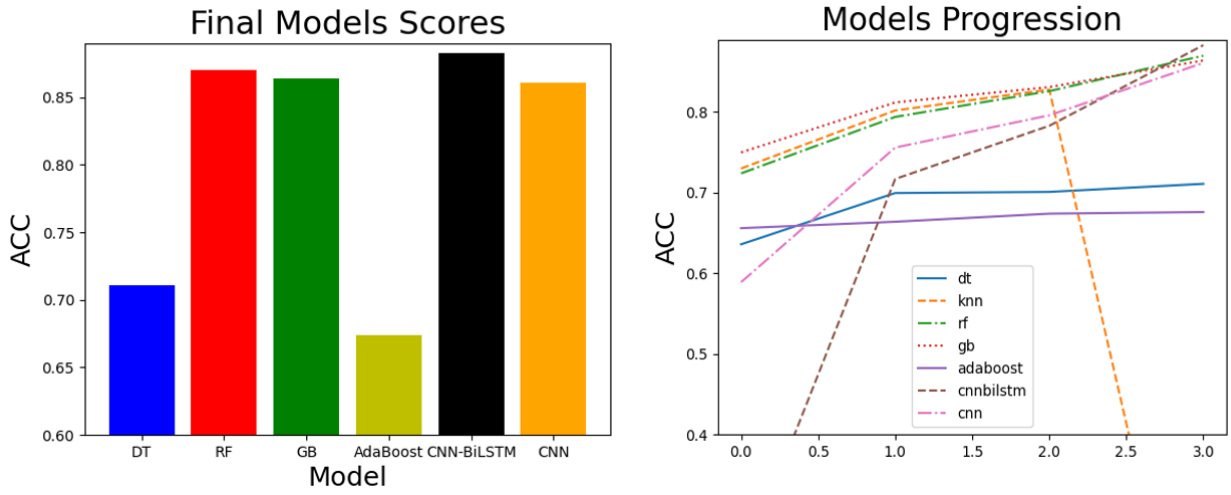

Figure 8: The histogram includes the accuracy of each final model with the whole input data. The line plot, represents the evolution of the accuracy of each model with the different dimension of the input data.

However, AdaBoost, does not seem to be as good since the model accuracy is very low and does not improve along the dimension change, as can be appreciated in Figure 8 . In particular, its poor prediction could be due to the noise of the input data, since AdaBoost is extremely sensitive to noisy data.

Furthermore, this claim is supported by the performance of Decision Tree, which is also particularly sensitive to data perturbations. Thus, these models tend to overfit and have a low accuracy in this situation. Therefore, if we wanted to improve the model generalization, we should improve the quality of the data; otherwise, our program would be limited to non

noise-sensitive models.

In the other hand, we have experienced a computational problem with the KNN algorithm, as the computing time increased with the dimension of the data, to the point that it was impossible to run it with the entire data set. this is due to its computational complexity and the large amount of memory required to store the training set before predicting the result. Therefore, to apply this algorithm with 900,000 input entries , a lot of time and RAM is necessary.

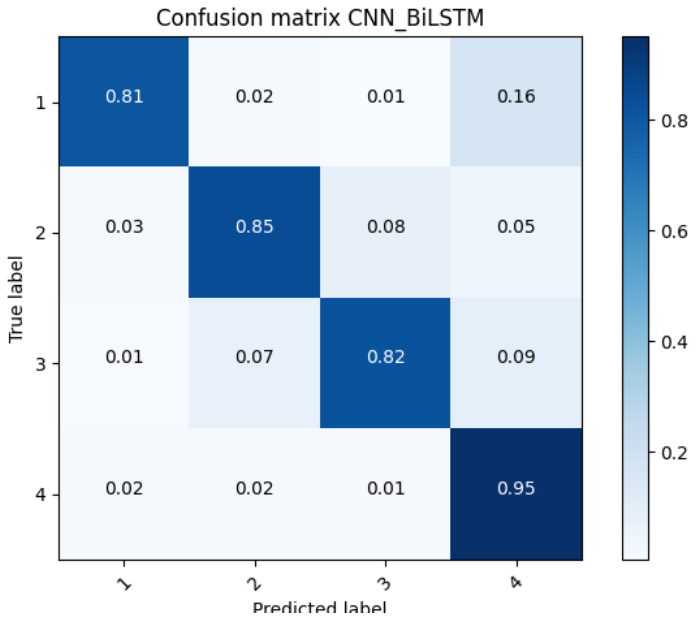

Figure 9: Confusion matrix of the final model with CNN-BiLSTM architecture.

Finally, we can see that CNN-BiLSTM is an architecture (Figure 10) that combines a bidirectional LSTM and a CNN generating a powerful model with high performance, as shown in Figure 9. CNNs (more information in Annexe 2) provide an efficient dense network that has little dependence on pre-processing data. BiLSTM-based models, on the other hand, provide better predictions, as they attempt to learn when to forget or not information by using gates.

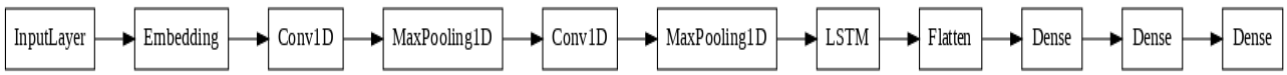

Figure 10: Layers of the CNN-BiLSTM architecture.

For that reason the CNN model does not predict as well as CNN-BiLSTM, as it lacks the strong effect BiLSTM has on the performance.

5

Conclusions and Futures Lines of Research

As expressed in the Introduction, the main goal of this project is to study if a Deep Learning based model is a robust approach for the prediction using Multiple Sequences Alignment as input. Therefore, we will use this section to state a personal but objective conclusion according to the results presented in the previous part of the report.

First of all, it is important to mention that from the very beginning this study has been challenging and motivating at the same time, for the main reason that there were few similar projects involving DL and MSA, so we had the opportunity to innovate and add our personal touch to the implementation. In addition, this work has given us the opportunity to strengthen our knowledge in Machine Learning and to acquire a perspective of Deep Learning as a powerful technique that can extend its application to different fields of study.

According to the principal objective of this study, we have given many reasons why DL could be a good solution to MSA problems, not only for he high quality of its prediction, but also for being the best solution despite the tough competition generated by other ML models like Gradient Boosting or Random Forest.

Nevertheless, having into account that the computational power of the devices employed to make this study and the time extension are very limited, we think that the performance of some of the models could be improved, and even that there could be other models which have a better accuracy for this decision-making problem.

Furthermore, we must highlight the importance of the Web application developed, since it enables us to easily apply the techniques studied in this thesis, and has allowed us to learn the basics of modern Web technologies.

As future work, MachineLAlign could have a real application in Bioinformatics and biological laboratories, but the efficiency of the tool would need to be improved by removing sequence length constraints and optimizing its execution time. In addition, to obtain better alignments, the call to other MSA tools could be replaced by the call to functions that implement each part of the alignment process (more information in Annex 1).

6

Conclusiones y Líneas Futuras

Tal y como se comentaba en la sección de Introducción, el objetivo principal de este proyecto era estudiar si un modelo basado en Aprendizaje Profundo es un enfoque robusto para la predicción clases para la Alineación de Secuencias Múltiples. Por lo tanto, utilizaremos este apartado para plantear una conclusión personal y objetiva de acuerdo con los resultados presentados en el apartado anterior.

En primer lugar, es importante mencionar que desde el principio, este trabajo, que podríamos considerar de investigación, ha sido desafiante y motivador al mismo tiempo, puesto que en la actualidad existen pocos proyectos similares que involucren tanto DL como MSA, por lo que tuvimos la oportunidad de innovar y agregar nuestro toque personal a la implementación. Además, este trabajo me ha brindado la oportunidad de asentar mi conocimiento en Machine Learning y adquirir una perspectiva del Deep Learning como una técnica poderosa que puede extender su aplicación a diferentes campos de estudio.

De acuerdo con el objetivo principal de este estudio, hemos dado muchas razones por las que DL podría ser una buena solución a los problemas de que supone Alinear Múltiples Secuencias, no solo por la alta calidad de su predicción, sino también por ser la mejor solución a pesar de la gran 'competencia' que han dado a lugar otros modelos de ML como Gradient Boosting o Random Forest.

Sin embargo, teniendo en cuenta que la potencia computacional de los dispositivos empleados para realizar este estudio y la extensión del tiempo eran bastante limitados, pensamos que el rendimiento de algunos de los modelos podría mejorarse e incluso podrían surgir modelos con una mayor precisión para este tipo de problemas de clasificación.

Además, MachineLAlign podría tener una aplicación real en los laboratorios bioinformáti-

cos y clínicos, pero para ello es necesario mejorar la eficiencia de la herramienta, eliminando las restricciones de longitud de secuencias y optimizando su tiempo de ejecución. Es más, para obtener mejores alineaciones, la llamada a otras herramientas de MSA podría substituirse por la llamada a funciones que implementan cada parte del proceso de alineación (más información en Anexe 1).

Por último, debemos destacar la importancia de la aplicación Web desarrollada, ya que nos permite aplicar fácilmente las técnicas de ML estudiadas en este TFG, y además, nos ha dado la posibilidad de comprender los fundamentos de las tecnologías Web modernas.

7 Annexes

7.1 Multiple Sequence Alignments

As described in the Introduction, **Multiple Sequence Alignment** is a problem derived from pairwise alignments, in which two or more sequences are optimally aligned, in order to identify commonly shared regions between biological sequences.

Mathematically, it can be defined as a problem with an input of *n* sequences Sn where n= 1, 2, 3, ..., i of length *L*, generating a matrix of size n X L, as it can be appreciated below.

$$\underbrace{\begin{bmatrix} S_{11} & S_{12} & \dots & S_{1L} \\ S_{21} & S_{22} & \dots & S_{2L} \\ \vdots & \vdots & \vdots & \vdots \\ S_{n1} & S_{n2} & \dots & S_{nL} \end{bmatrix}}_{\mathbf{s}}$$

Thus, sequences can be aligned by inserting gaps until the desired alignment is reached, as we can observe in Figure 11.

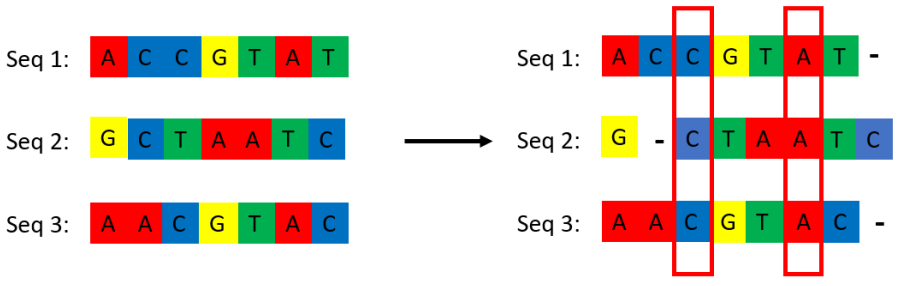

Figure 11: Example of alignment procedure with DNA sequences, where each color represents a nucleotide.

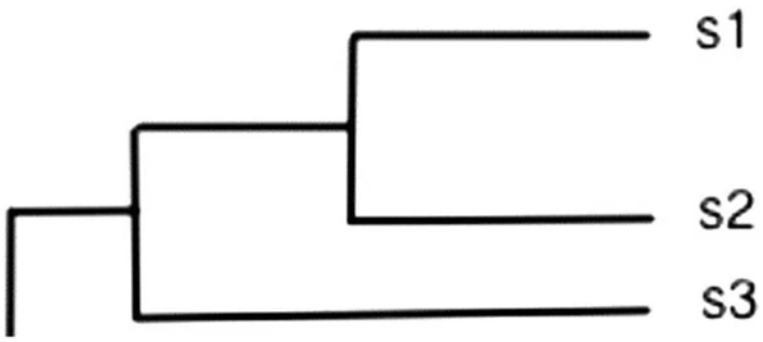

Figure 12: Order of the progressive alignment method. Taken from [27]

7.1.1 Computational Approaches

As we can imagine, the goal is to reach an optimal arrangement which implies solving a problem of NP-hard (Non-deterministic Polynomial) complexity, since the the bigger the number of sequences (n) and their lengths (L), more increases the computational costs, specifically the memory and time required.

In order to reduce these drawbacks, different algorithms and heuristic have been applied toward MSA, some of which will be explained as follows:

- **Dynamic Programming (exact methods)**

  This method is commonly used for pairwise sequence alignments, but it can be a difficult and time-consuming task for MSA. The complexity increases quadratically with respect to the number of sequences (n) and their lengths (L).

  In the case where sequences are proteins, this technique consists of employing a gap penalty and a substitution matrix to assign scores based on the similarity of chemical properties and the probability of mutation.

  In contrast, for nucleotide, the gap penalty is similar, but the substitution matrix only takes into account identical matches or mismatches between nucleotide pairs.

- **Progressive Alignment**

  In this case, two sequences are chosen and aligned, then other sequence is aligned with the previous ones and the process is repeated until the end (see Figure 12).

  Commonly, this strategy includes five phases, as shown in Figure 13:

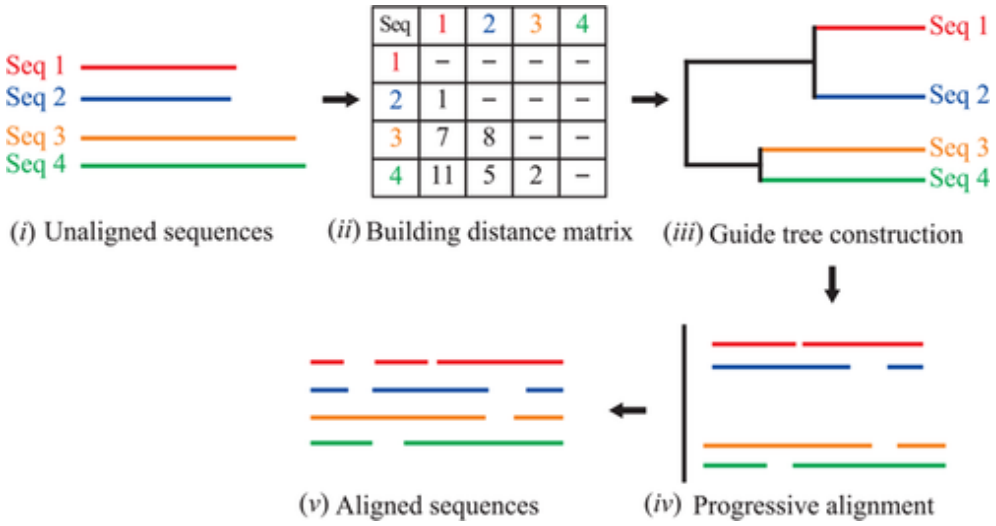

Figure 13: Phases of the progressive alignment method. Taken from [16]

$$\mathbf{P}(x_i \sim y_j \in a^* | x, y) = \sum_{a \in A} \mathbf{P}(a|x,y)\mathbf{1}\{x_i \sim y_j \in a\}$$

Figure 14: Formula of the posterior probability matrix calculation.

1. Posterior probability matrix calculation

This step consists on computing the updated probability of an event occurring just after a new decision has been made (Bayesian statistics). It means, "is the probability of event A occurring given that event B has occurred" as expressed in Bayes' theorem [28].

Mathematically, having sequence x and y with residues xi and yi, and a* as their "true" biological alignment, it can using the following definition (Figure 14) where A refers to all possibles alignments for x and y, and 1{*expression*} implies that the function will return 1 if the expression is True and 0 if False. Thus, posterior probability "represents the probability of xi aligned to yi in th true alignment a*" [29].

For Multiple Sequence Alignment we can consider different methods for getting this matrix, so we will presents then the most frequently used:

- Pair-Hiden Maekov Model (HMM)
- Partition function

- RMS of pair-HMM and partition function
- RMS of pair-HMM, partition function and random HMM

2. Calculation of the distance matrix having sequence x and y, we calculate the distance between each pair of sequences.

3. Guide tree construction At this part it is constructed a data structure to determine the relationship between two sequences or profiles. It is based on the distance matrix previously generated.

In this case, we will consider two options:

1) Unweighted Pair Group Method with Arithmetic mean (UPGMA)

2) Weighted Pair Group Method with Averaging (WPGMA)

4. Consistency transformation: Consists on the posterior probability relaxation.

5. Refinement: Correct errors.

- **Iterative Refinement Methods Alignment**

  This method is similar to previous one (progressive alignment), but in this case there is a process of re-evaluation of the alignment at each step. In other words, in this algorithm a suboptimal solution is calculated by progressive alignment in order to perform a subsequent modification by dynamic programming until convergence is reached.

  The advantage of this process is that if an error is made, it can be resolved before computing the next step. Muscle is one of the existing tools that uses this method.

- **Hidden Markov Models**

  A HMM consists of a probabilistic model of MSA where each column of elements (amino acids) to be aligned is represented by an state, the frequency distribution of elements, and insertions and deletions are represented by other states, as indicated by Mount, David W. [30].

  In this method, to match a given sequence, the model transitions from state to state in a Markov chain. Thus, the next paired element is chosen from each state, storing its

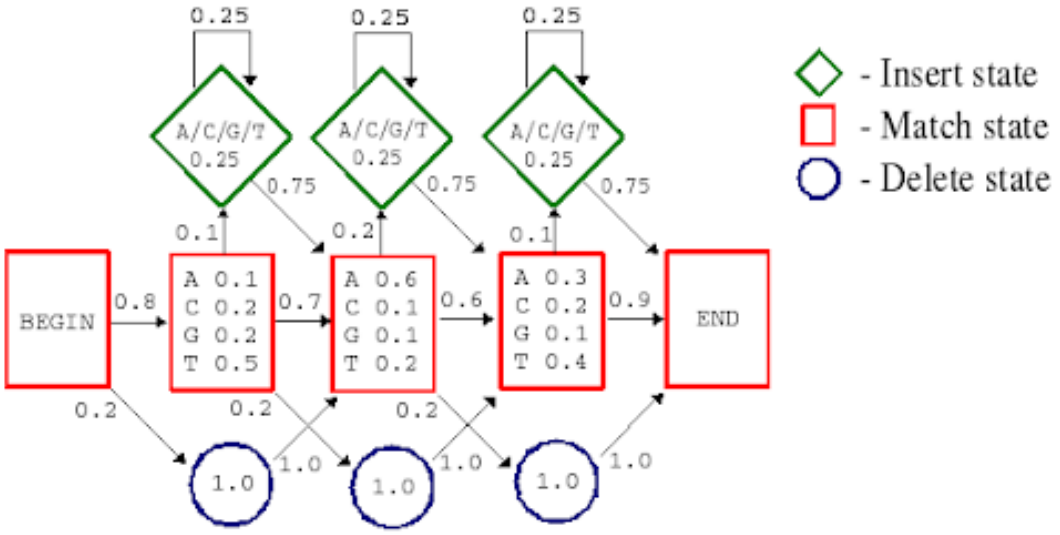

Figure 15: Representation of an HMM. Taken from Hidden Markov models (HMM)[31]

frequency and the transition probability (going to that state after a previous one). Finally, the probability of the sequence is calculated by multiplying the state and transition probabilities.

In summary, we can define HMM as a network of interconnected states (insert, match and delete) that emit an output symbol. The states are composed of symbol emission probabilities, in other words, the frequency of emission of each symbol from a state, and the state transition probabilities. The schema of a HMM is presented in Figure 15.

The lack of information about the value of a specific state defines the hidden nature of HMM, which is shown by the probability distribution over all possible values.

7.1.2 Score Metrics

There are several method for scoring MSA, but then I will describe the most common ones:

- **Percentage of totally conserved columns (TC)**: It represents the total number of columns whose rows share the same amino acid or nucleotide.
- **Percentage of non-gaps**: this value indicates the number of elements in the alignment which are not gaps.

- **Minimum Entropy (H(s))**: It is a metric used to show the level of divergence among the residues in a column.

- Using a substitution matrix: A determined matrix is employed to calculate the value that classifies the sets of aligned sequences as more or less optimal. There are different approaches:

  - **Sum of pairs (SP)**: Given a sequence alignment, the possible combinations of pairs by column are selected and the sum of the values provided by the substitution matrix is computed. Therefore, the output is a final sum obtained from the sum of each column.

  - **Star**: This method can be divided into two parts, the first one to search for the most repeated element in each column, and the second one to calculate the distances of the remaining symbols to the most obtained symbol in the first part, using a substitution matrix.

  - **Weighted sum of pairs**: This algorithm is similar to SP, but in this case weights are added to multiply each value of the substitution matrix.

- **Using structural information**: This metric is the least common and relies on structural information to perform the best alignment. It starts from evaluating the structural similarity of the sequences, measured in Root-Means-squared-Deviation (RMSD), and then map all conformational motifs into the alignment. Therefore, if sequences have similar structures, the alignment will be more accurate. [32]

7.2 Deep Learning

Deep learning [33] can be defined as an area of data science research pertaining to Machine Learning algorithms. It is inspired by the structure of the human brain, in which neurons participate transmitting signals from nerve impulses to muscles. The process can be considered as unsupervised learning, which does not rely on previously tagged data, but is able to learn by itself without any external information. Therefore, we can sum up the purposes of this methodology in three points:

- Learn through experience (training).

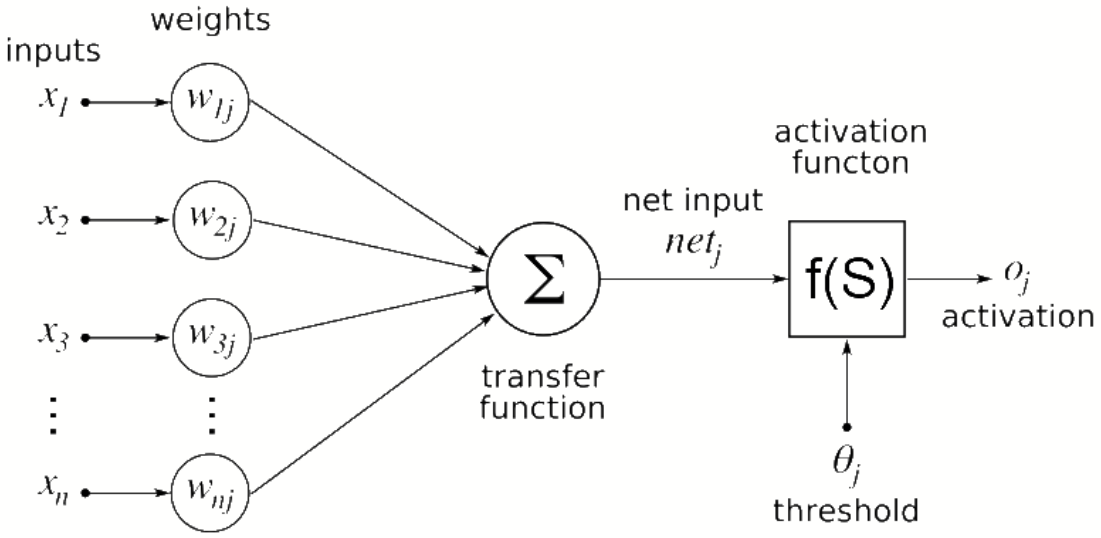

Figure 16: Representation of the neural network architecture. Taken from *How does deep learning work?* [34]

- Obtain information from the input data.
- Apply what has been learned by recognizing patterns.

These models are organized in layers, composed of artificial neurons that are connected to each other to share relevant information. This allows to determine an output or output value. In addition, each of the connections between layers have a weight associated with it, which indicate the importance of the input value. The structure of a DNN can be observed in Figure 16.

7.2.1 Activation Function

Each neuron has an activation function whereby an output is calculated from a set of inputs. It is used to normalize the output and to determine the result (accuracy) of the model performance. These functions can be classified as (showed at Figure 17):

1) Linear functions: It is also Known as entity, where the input is to the output.

2) Nonlinear functions

> Step function: It is based on thresholds, so it returns 1 for values greater than 0, and 0 for the rest. It is used to classify or determine categorical outputs.

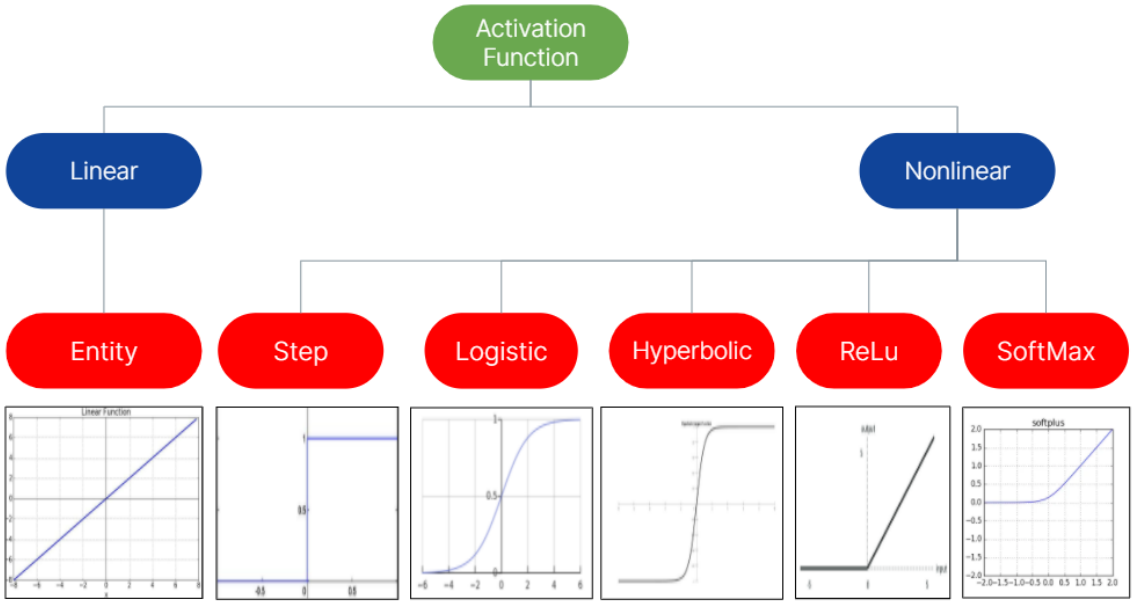

Figure 17: Diagram showing the most common activation functions.

> Sigmoid function: The output is interpreted as a probability where it will be 0 if the input has values less than 0, 0.5 if it has a value of 0 and 1 for the rest of cases.

> Hyperbolic function: The output values are in rage of -1 to 1, with high values tending to 1 asymptotically and low values to -1.

> ReLU (Rectified Linear Unit) function: With this function, negative input values have a result of 0 and for positive ones it returns the same value as the input. It is one of the most used functions because of its high-speed learning capability. In practice, it is applied for image analysis.

> Softmax function: It is used to classify data between multiple classes by assigning probabilities to each of them. For example, if we introduce the image of a bone, applying this function, the network would return 30% of ulna bone, 20% of femur bone or 50% of humerus bone.

7.2.2 Types of architectures

- Deep Neural Networks

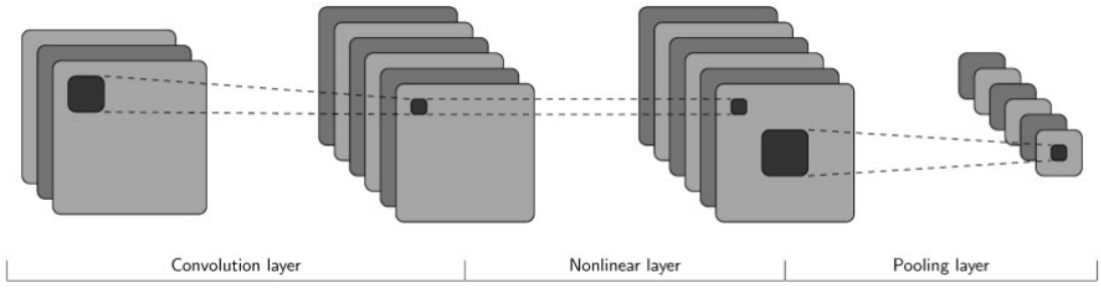

Figure 18: Structure of Convolutional Neural Networks with the mentioned layers. Taken from Deep learning in Bioinformatics [35].

It has a basic structure consisting of an input layer, several hidden layers and a final output layer. Given the input data, the output values are calculated sequentially across the layers of the network. At each layer, the input vector corresponding to the output values of each unit of the previous layer is multiplied by the vector of weights of each unit of the current layer to obtain a weighted sum, to which can be applied any of the nonlinear functions mentioned above [35].

DNN architectures are commonly used for analyzing high-dimensional data as they have the capacity to find out highly abstract patterns and correlations previously unknown.

- Convolutional Neural Networks

CNNs are a type of Neural Network that are inspired by the functioning of the human's visual cortex, a powerful processing system. It bases on the ideas of local connectivity, localization invariance, and local transition invariance. Thus, they make use of convolution, non-linear and pooling layers, as it is can be noted in Figure 18.

Local weighted sums called feature maps are obtained to discover highly correlated sub-regions of data. At the same time, in each convolution layer, filters are calculated and applied repeatedly throughout the set, thus indirectly improving training efficiency by parameter reduction.

Then, nonlinear layers are able to augment the properties of the feature maps and prepare them for the final layer. The latter, called pooling layer, participates in a maximum or average subsampling of the non-overlapping regions in the feature maps. It allows to to handle somewhat different but semantically similar features and thus add local characteristics to identify

more complex patterns.

Nowadays, CNNs are one of the most widely used architectures due to their capacity to analyze dimensional data, but also because of their wide range of applications. For instance, they have great achievements in the area of biomedical imaging recognition, signal processing or data classification. For this reason, we have decided to make use of this type of structures.

- Recurrent Neural Networks

They process sequential input data and perform the calculations in the hidden units where there are recurring or cyclic connections.

Therefore, the information is stored in state vectors (hidden units), while the output for the current input is calculated taking into account all the previous inputs whose values had been stored in state vectors. In addition, Bidirectional RNN have been designed to allow new and old inputs to determine the output. In conclusion, these architectures are widely employed by researchers in different fields such as Natural Language Processing or language translation.

Finally, there are many emergent architectures that propose alternative models and perform in a similar way. We can highlight DST-NN, designed to learn multidimensional output targets through progressive refinement, or CAE which is defined to learn hierarchical representations of data. However, DL has a great future in many areas of study, so new and better approaches are to be expected.

Bibliographic references

[1] "What is Artificial Intelligence? How Does AI Work? | Built In." [Online]. Available: https://builtin.com/artificial-intelligence

[2] S. R. Eddy, "What is a hidden Markov model?" pp. 1315–1316, oct 2004. [Online]. Available: http://www.nature.com/naturebiotechnology

[3] "Bioinformatics made easy: Bioinformatics: Main Applications Of Multiple sequence Alignment." [Online]. Available: http://bioinformatics-made-easy.blogspot.com/2009/12/bioinformatics-main-applications-of.html

[4] "kuangmeng/DLPAlign: Deep Learning based Progressive Alignment for Multiple Protein Sequences." [Online]. Available: https://github.com/kuangmeng/DLPAlign

[5] "Scikit-learn: Machine Learning in Python | The Journal of Machine Learning Research." [Online]. Available: https://dl.acm.org/doi/10.5555/1953048.2078195

[6] "GitHub - benhid/pyMSA: Scoring multiple sequence alignments with Python." [Online]. Available: https://github.com/benhid/pyMSA

[7] "TensorFlow." [Online]. Available: https://www.tensorflow.org/

[8] "What is Bootstrap." [Online]. Available: https://www.w3schools.com/whatis/whatis_bootstrap.asp

[9] "ClustalW2 < Multiple Sequence Alignment < EMBL-EBI." [Online]. Available: https://www.ebi.ac.uk/Tools/msa/clustalw2/

[10] "MAFFT - a multiple sequence alignment program." [Online]. Available: https://mafft.cbrc.jp/alignment/software/

[11] "T-COFFEE Multiple Sequence Alignment Server." [Online]. Available: http://tcoffee.crg.cat/

[12] "Clustal Omega < Multiple Sequence Alignment < EMBL-EBI." [Online]. Available: https://www.ebi.ac.uk/Tools/msa/clustalo/

[13] C. Notredame, D. G. Higgins, and J. Heringa, "T-coffee: A novel method for fast and accurate multiple sequence alignment," *Journal of Molecular Biology*, vol. 302, no. 1, pp. 205–217, sep 2000.

[14] D. F. Feng and R. F. Doolittle, "Progressive sequence alignment as a prerequisitetto correct phylogenetic trees," *Journal of Molecular Evolution*, vol. 25, no. 4, pp. 351–360, aug 1987. [Online]. Available: https://link.springer.com/article/10.1007/BF02603120

[15] J. D. Thompson, D. G. Higgins+, and T. J. Gibson, "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," Tech. Rep., 1994.

[16] "ClustalW based progressive alignment. | Download Scientific Diagram." [Online]. Available: https://www.researchgate.net/figure/ClustalW-based-progressive-alignment_fig2_332443283

[17] R. C. Edgar, "MUSCLE: Multiple sequence alignment with high accuracy and high throughput," *Nucleic Acids Research*, vol. 32, no. 5, pp. 1792–1797, 2004. [Online]. Available: /pmc/articles/PMC390337//pmc/articles/PMC390337/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC390337/

[18] "A Beginner's Guide to Deep Reinforcement Learning | Pathmind." [Online]. Available: https://wiki.pathmind.com/deep-reinforcement-learning

[19] R. K. Ramakrishnan, J. Singh, and M. Blanchette, "RLALIGN: A reinforcement learning approach for multiple sequence alignment," in *Proceedings - 2018 IEEE 18th International Conference on Bioinformatics and Bioengineering, BIBE 2018*. Institute of Electrical and Electronics Engineers Inc., dec 2018, pp. 61–66.

[20] I. G. Mircea, I. Bocicor, and G. Czibula, "A reinforcement learning based approach to multiple sequence alignment," in *Advances in Intelligent Systems and Computing*, vol. 634. Springer Verlag, 2018, pp. 54–70. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-62524-9_6

[21] M. Kuang, "DLPAlign: A deep learning based progressive alignment for multiple protein sequences," p. 2020.07.16.207951, jul 2020. [Online]. Available: https://doi.org/10.1101/2020.07.16.207951

[22] C. Mirabello and B. Wallner, "rawMSA: End-to-end Deep Learning using raw Multiple Sequence Alignments," *PLOS ONE*, vol. 14, no. 8, p. e0220182, aug 2019. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0220182

[23] C. B. Do, M. S. Mahabhashyam, M. Brudno, and S. Batzoglou, "ProbCons: Probabilistic consistency-based multiple sequence alignment," *Genome Research*, vol. 15, no. 2, pp. 330–340, 2005. [Online]. Available: /pmc/articles/PMC546535//pmc/articles/PMC546535/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC546535/

[24] "Decision Trees in Machine Learning | by Prashant Gupta | Towards Data Science." [Online]. Available: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

[25] "Understanding AdaBoost. Anyone starting to learn Boosting... | by Akash Desarda | Towards Data Science." [Online]. Available: https://towardsdatascience.com/understanding-adaboost-2f94f22d5bfe

[26] "sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.24.2 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[27] R. Jafari, M. M. Javidi, and M. Kuchaki Rafsanjani, "Using deep reinforcement learning approach for solving the multiple sequence alignment problem," *SN Applied Sciences*, vol. 1, no. 6, p. 592, jun 2019. [Online]. Available: https://doi.org/10.1007/s42452-019-0611-4

[28] "Posterior Probability Definition." [Online]. Available: https://www.investopedia.com/terms/p/posterior-probability.asp

[29] U. Roshan and D. R. Livesay, "Probalign: Multiple sequence alignment using partition function posterior probabilities," *Bioinformatics*, vol. 22, no. 22, pp. 2715–2721, nov 2006. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/16954142/

[30] D. W. Mount, "Using hidden Markov models to align multiple sequences," *Cold Spring Harbor Protocols*, vol. 4, no. 7, 2009. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/20147223/

[31] "Assignment 2." [Online]. Available: http://www.cbs.dtu.dk/$\sim$kj/bioinfo_assign2.html

[32] "How to score an MSA — jMSA 1.0-SNAPSHOT documentation." [Online]. Available: https://jmsa.readthedocs.io/en/latest/how_to_score.html

[33] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," pp. 436–444, may 2015. [Online]. Available: http://colah.github.io/

[34] "Packt Subscription | Learn more for less." [Online]. Available: https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781789802993/1/ch01lvl1sec11/how-does-deep-learning-work

[35] S. Min, B. Lee, and S. Yoon, "Deep learning in bioinformatics," pp. 851–869, sep 2017. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/27473064/

[36] "MUSCLE < Multiple Sequence Alignment < EMBL-EBI." [Online]. Available: https://www.ebi.ac.uk/Tools/msa/muscle/

[37] J. D. Thompson, B. Linard, O. Lecompte, and O. Poch, "A Comprehensive Benchmark Study of Multiple Sequence Alignment Methods: Current Challenges and Future Perspectives," *PLoS ONE*, vol. 6, no. 3, p. e18093, mar 2011. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0018093

[38] "T-COFFEE Multiple Sequence Alignment Server." [Online]. Available: http://tcoffee.crg.cat/

[39] K. Katoh, K. Misawa, K. I. Kuma, and T. Miyata, "MAFFT: A novel method for rapid multiple sequence alignment based on fast Fourier transform," *Nucleic Acids Research*, vol. 30, no. 14, pp. 3059–3066, jul 2002. [Online]. Available: /pmc/articles/PMC135756//pmc/articles/PMC135756/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC135756/

[40] N. H. Shah and J. D. Tenenbaum, "The coming age of data-driven medicine: translational bioinformatics' next frontier," *Journal of the American Medical Informatics Association*, vol. 19, no. e1, pp. e2–e4, jun 2012. [Online]. Available: https://academic.oup.com/jamia/article-lookup/doi/10.1136/amiajnl-2012-000969

[41] W. Liu, B. Schmidt, G. Voss, and W. Müller-Wittig, "Streaming algorithms for biological sequence alignment on GPUs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 9, pp. 1270–1281, 2007.

[42] "Multiple sequence alignment." [Online]. Available: https://www.ncbi.nlm.nih.gov/CBBresearch/Przytycka/download/lectures/PCB_Lect05_Multip_Align.pdf

[43] "Multiple sequence alignment - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Multiple_sequence_alignment#Dynamic_programming

[44] "Multiple Sequence Alignment." [Online]. Available: https://bioinf.comav.upv.es/courses/biotech3/theory/multiple.html

[45] "Msa." [Online]. Available: https://informatica.cv.uma.es/pluginfile.php/354914/mod_resource/content/0/OmicsMasterTutorial.MSA.pdf

[46] "NP-Hard Problem – from Wolfram MathWorld." [Online]. Available: https://mathworld.wolfram.com/NP-HardProblem.html

[47] "Alineamientos de múltiples secuencias. Rodrigo Santamaría - PDF Free Download." [Online]. Available: https://docplayer.es/40278110-Alineamientos-de-multiples-secuencias-rodrigo-santamaria.html

[48] "CNN BiLSTM Explained | Papers With Code." [Online]. Available: https://paperswithcode.com/method/cnn-bilstm

[49] "Keras model with sequence feature columns fails to convert to estimator · Issue 32341 · tensorflow/tensorflow." [Online]. Available: https://github.com/tensorflow/tensorflow/issues/32341

[50] H. Fukuda and K. Tomii, "DeepECA: An end-to-end learning framework for protein contact prediction from a multiple sequence alignment," *BMC Bioinformatics*, vol. 21,

no. 1, p. 10, jan 2020. [Online]. Available: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-019-3190-x

[51] J. Daugelaite, A. O' Driscoll, and R. D. Sleator, "An Overview of Multiple Sequence Alignments and Cloud Computing in Bioinformatics," *ISRN Biomathematics*, vol. 2013, pp. 1–14, aug 2013.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA