OpenVaccine: COVID-19 mRNA Vaccine Degradation Prediction

Imane MESSAK

June 20, 2021

1 Introduction

Coronavirus disease 2019 (COVID-19) is caused by a novel coronavirus known as Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2), which is associated with several fatal cases around the world. The rapid spread of this pathogen and the growing number of cases highlight the urgent development of vaccines. This urgent need for vaccines against the disease due to the ongoing SARS-CoV-2 pandemic has resulted in several vaccines that are still under study. Among all the approaches, a messenger RNA (mRNA) vaccine has emerged as a rapid and versatile platform to respond rapidly to this challenge. These mRNA vaccines have taken the lead as the fastest vaccine candidates for COVID-19, but currently they face key potential limitations []. Stanford University has discovered that one of the biggest challenges today is how to design super stable messenger RNA (mRNA) molecules. Conventional vaccines (like your seasonal flu shots) are packaged in disposable syringes and shipped under refrigeration around the world, but this is currently not possible for mRNA vaccines [1]. In fact, researchers have observed that RNA molecules tend to degrade spontaneously. This is a serious limitation - a single cut can render the mRNA vaccine useless. Currently, little is known about the details of where in the backbone of a given RNA is most likely to be affected. Without this knowledge, current COVID-19 mRNA vaccines must be prepared and shipped under intense refrigeration, and they are unlikely to reach more than a tiny fraction of humans on the planet unless they are they cannot be stabilized.

The objective is therefore to create a model that can predict the rate of degradation of each base of each mRNA sequence of a subset from a dataset containing 3029 RNA sequences. We have three qualitative and structural pieces of information describing mRNA sequences: sequence, base pairing and loop prediction. We are therefore going to create three prediction models by taking only the sequence as an explanatory variable at first, then a second model which will take into account the sequence and the base pairing information and finally a third model which will take into account the three explanatory variables. Then we will compare these three models to see which is better and especially if the introduction of structural information improves the prediction.

The code and results can be checked on the GitHub link: `https://github.com/imanemessak/OpenVaccine-COVID-19-mRNA-Vaccine-Degradation-Prediction`

2 Material & Methods

2.1 Data Description

Stanford scientists have data on 3029 RNA sequences of length 107. For technical reasons, measurements cannot be performed on the final bases of these RNA sequences, so we have experimental data (ground truth) under 5 conditions “reactivity”, “deg_Mg_ph10”, “deg_Mg_50c”, “deg_ph10”, “deg_50c” for the first 68 bases. 629 of these 3029 sequences were distributed for a public test in order to allow continuous evaluation throughout the competition, in the public ranking. These sequences, in test.json, were further filtered on the basis of three criteria ”sequence”, ”structure” and ”predicted_loop” to ensure that this subset is not dominated by a large cluster of molecules of RNA with poor data, which could skew the public ranking. The remaining 2400 sequences for which we have data are in train.json. For our final and most important score (Private Leaderboard), Stanford scientists are performing measurements on 3,005 new RNAs, which have lengths a little longer than 130 bases. For these data, we would expect to have measurements for the first 91 bases, again missing the ends of the RNA. These sequences constitute another 3005 of the 3634 test.json sequences. [1]

2.2 Pipeline

Before creating our model, we cleaned and processed our data to gain insight into the distribution of degradation values. To begin with we removed all the sequences that had a signal/noise ratio less than 1. Indeed a ratio less than 1 means that there were errors at the experimental level, and these samples can lead to a bad prediction. Then we looked at the redundancy of our sequences thanks to the CD_hit web server [2]. Indeed, if the identity rate between the different sequences is too high and if there are too many similar sequences, our model will learn badly and therefore badly predict. As an

output, we get a file which splits our game into several clusters. A cluster means that the sequences are considered to be similar and there is an associated identity rate. This redundancy can be treated by weighting the sequences with "sample_weight" on *keras*.

2.3 Model construction methods

Different types of neural network exist, for our prediction, we used a GRU (Gated Recurrent Unit) type neural network. GRU is a variant of RNN type networks. RNN is the most suitable type of network for our type of predictor, however this network has limitations. Indeed an RNN uses the method of the descent of the gradient in order to update the weights between its neurons. However, as learning progresses, updating the pea does not do very well and therefore the model learns poorly and may forget old data. The memory of an RNN is therefore rather short. For this, we used GRU, which has a better architecture and which can solve short and long term memory problems. For the loss, we have implemented the MCRMSE (mean columnwise root mean square error) function. The MCRMSE is an average of all RMSE values for each column in the dataset. And the RMSE, is an average of the difference between the observed values and the predicted values. Then we played on several parameters such as the optimization function, activation, the number of layers and the number of epochs. Indeed, for the optimization functions, we tested different algorithms that are considered one of the two most powerful optimizers used in deep learning, which are RMSprop (Root Mean Square Propagation) and Adam. For the activation functions we tried the most efficient: ReLU (the most efficient) and Sigmoid.

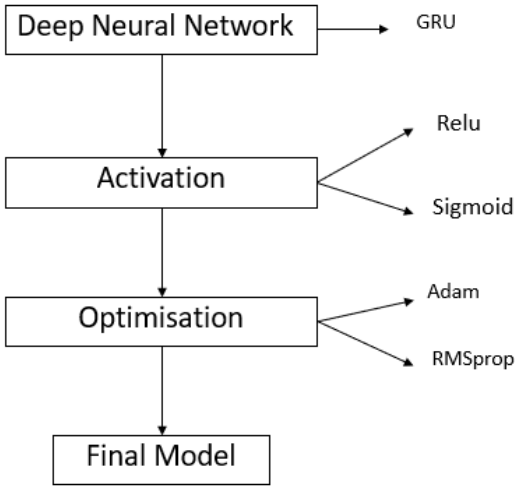

Figure 1: Diagram of our pipeline

3 Analysis & Conception

3.1 Cleaning & Preparing the dataset

Our data cleaning consists of removing all sequences whose signal/noise ratio is less than 1. After deleting these sequences, we obtain a set of 2097 samples. Then, we used the CD_hit web server to find if our sequences are redundant.

Data preparation consists of encoding our qualitative variables "sequence", "structure" and "predicted_loop" using one-hot encoding to obtain the result below. our dataset is now contains 22 columns.

One_hot seq	One hot strucure	encoding predicted loop
[[[1, 0, 0, 0]], [[1, 0, 0, 0]], [[0, 1, 0, 0]...	[[[1, 0, 0]], [[1, 0, 0]], [[1, 0, 0]], [[1, 0...	[[[1, 0, 0, 0, 0, 0, 0]], [[1, 0, 0, 0, 0, 0, ...
[[[1, 0, 0, 0]], [[1, 0, 0, 0]], [[0, 1, 0, 0]...	[[[1, 0, 0]], [[1, 0, 0]], [[1, 0, 0]], [[1, 0...	[[[1, 0, 0, 0, 0, 0, 0]], [[1, 0, 0, 0, 0, 0, ...

Figure 2: The encoding of the train data

3.2 Exploratory analysis

The exploratory analysis consists of studying the 3 qualitative variables with respect to the 5 conditions "reactivity", "deg_Mg_ph10", "deg_Mg_50c", "deg_ph10", "deg_50c".

- Representation of the sequence data

We looked for the maximum value of the 5 conditions for each mRNA sequences, then we associated it with a base, then we counted the number of each base.

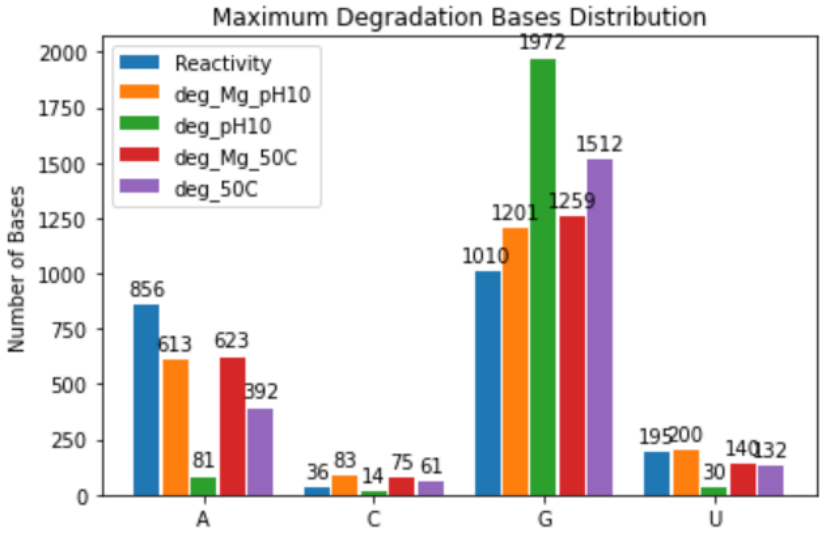

Figure 3: This frog was uploaded via the file-tree menu.

We see that the base G is the base which has the highest degradation rate, so we can deduce that the base G is the most susceptible to degradation for all the degradation variables followed by the base A.

- Representation of the structural data

Same as before, we looked for the maximum value of the 5 conditions for each mRNA sequences, then we associated it with a structure, then we counted the number of each structure.

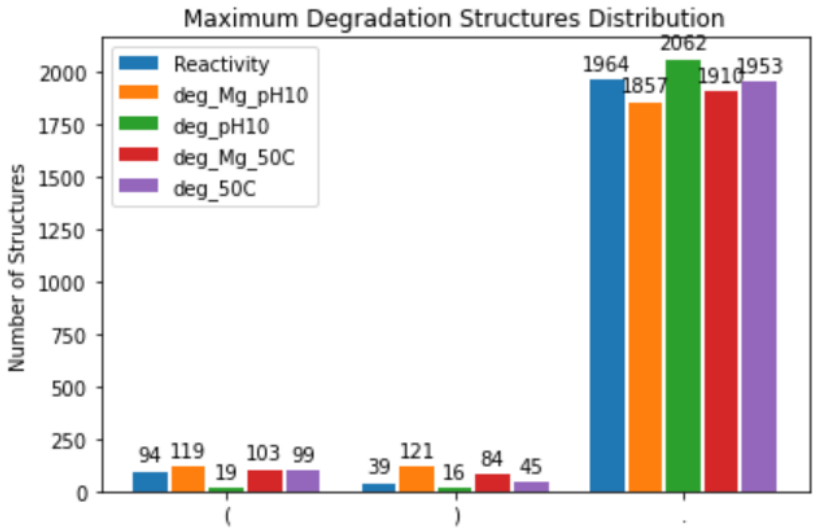

Figure 4: This frog was uploaded via the file-tree menu.

From the plot of the distribution of structures, we see that the unpaired bases are the most susceptible to degradation.

- Representation of the loop prediction

We looked for the maximum value of the 5 conditions for each mRNA sequences, then we associated it with a type of loops, then we counted the number of each type of loops.

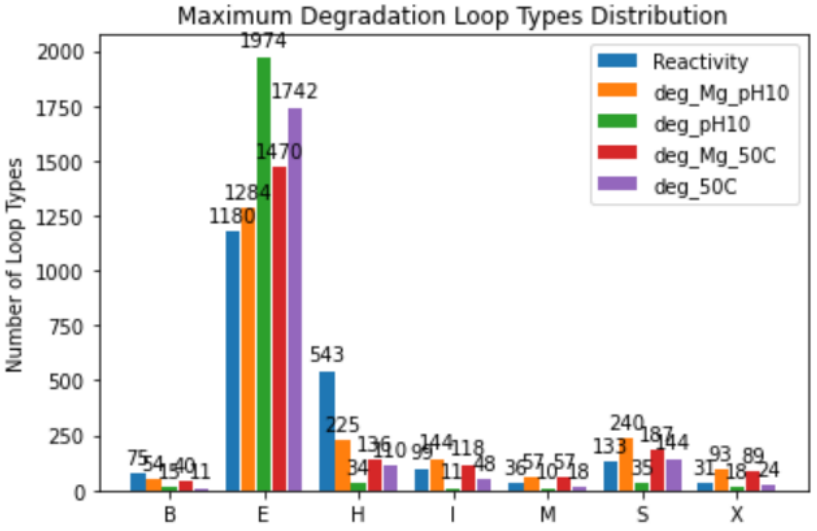

Figure 5: This frog was uploaded via the file-tree menu.

For the plot of the distribution of the types of loops, it can be seen that the E loops are the most susceptible to degradation.

3.3 Parameters

After choosing the activation methods, We tested for each activation function, an optimizer and different numbers of layers. We tried with 1, 2 then 3 layers and initially we started with a number of epochs of 45, then we increased it to 75, choosing a batch size of 64.

4 Results

4.1 The best selected prediction models

- One variable: Sequence

Parameters	Loss	Val_loss
Adam		
Relu		
3 Layres		
75 epoch	0.2214	0.2653

Table 1: Result of model 1 predicted from sequence information

- Two variables: Sequence + Structure

Parameters	Loss	Val_loss
Adam		
Relu		
3 Layres		
75 epoch	0.2018	0.2352

Table 2: Result of model 2 predicted from sequence and structure information

- Three variables: Sequence + Structure + Loop

Parameters	Loss	Val_loss
Adam		
Relu		
3 Layres		
75 epoch	0.1861	0.2323

Table 3: Result of model 3 predicted from sequence, structure and loop prediction

From the tables above, we therefore notice that by adding additional information (structure and / or prediction of loops) the value of the loss and of the val_loss decreases, and in all cases the value of the loss remains below the value of val_loss.

In the case of the first model we see a difference between loss and val_loss of 0.0445. This difference shows an overfitting of our model. However, this model was predicted from sequence information only. When we look at our second model, we observe that the value of loss and val_loss decrease and the difference between the two is of the order of 0.033. In this prediction model, the overfitting is much lower but the value of loss and val_loss remains quite high. Finally when we look at the third model, only the value of the loss decreases and the gap increases. We therefore return to overfitting. The introduction of observed structure information alone (model 2) improved the prediction of our model, but the introduction of predicted structure information does not improve the model and clearly increases overfitting. So the model predicts poorly from predicted information.

5 Conclusion

To conclude after having tested different models, we can conclude that the best prediction model that we obtained is model 2. In fact, it does not seem to present any overfitting (or even a slight overfitting) and at low values of loss and val_loss but not excellent. Although our model is not very efficient, we notice that our distributions of the most degradable bases, structures and loop predictions coincide well with those of the observed data.

Bibliography

[1] Kaggle: Openvacine competition. http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/.

[2] Cd-hit suite: a web server for clustering and comparing biological sequences.

Appendix A

Other Results

Parameters	Loss	Val_loss
Adam Relu 1 variable Batch 64 One layer 45 epochs	0.2192	0.2528
Adam Sigmoid 1 variable Batch 64 One layer	0.3092	0.3152
RMSprop Relu 1 variable Batch 64 One layer	0.2419	0.2561
RMSprop Sigmoid 1 variable Batch 64 One layer 45 epochs	0.3086	0.3196
Adam Relu 2 variables Batch 64 One layer 45 epochs	0.2192	0.2528
Adam Sigmoid 2 variables Batch 64 One layer 45 epochs	0.3092	0.3152
RMSprop Relu 2 variables Batch 64 One layer 45 epochs	0.2419	0.2561
RMSprop Sigmoid	0.3086	0.3196

2 variables Batch 64 One layer		
Adam Relu 3 variables Batch 64 One layer	**0.2192**	**0.2528**
Adam Sigmoid 3 variables Batch 64 One layer	**0.3092**	**0.3152**
RMSprop Relu 3 variables Batch 64 One layer	**0.2317**	**0.2538**
RMSprop Sigmoid 3 variables Batch 64 One layer	**0.3115**	**0.3154**
Adam Relu 1 variable Batch 64 2 layers	**0.2083**	**0.2404**
Adam Sigmoid 1 variable Batch 64 2 layers	**0.2999**	**0.3079**
RMSprop Relu 1 variable Batch 64 2 layers	**0.2089**	**0.2409**
RMSprop Sigmoid 1 variable Batch 64 2 layers	**0.3003**	**0.3090**

Adam Relu 2 variables Batch 64 2 layers	**0.2083**	**0.2404**
Adam Sigmoid 2 variables Batch 64 2 layers 45 epochs	**0.2999**	**0.3079**
RMSprop Relu 2 variables Batch 64 2 layers	**0.6491**	**0.6389**
RMSprop Sigmoid 2 variables Batch 64 2 layers	**0.3011**	**0.3091**
Adam Relu 3 variables Batch 64 2 layers	**0.2083**	**0.2404**
Adam Sigmoid 3 variables Batch 64 2 layers	**0.2999**	**0.3079**
SGD Relu 3 variables Batch 64 2 layers	**0.4436**	**0.4280**
RMSprop Relu 3 variables Batch 64 2 layers	**0.6497**	**0.6389**
RMSprop Sigmoid 3 variables	**0.3036**	**0.3090**

Batch 64 2 layers Epochs		
Adam Relu 1 variable Batch 64 3 layers	**0.2180**	**0.2412**
Adam Sigmoid 1 variable Batch 64 3 layers	**0.3033**	**0.3074**
RMSprop Relu 1 variable Batch 64 3 layers 45 epochs	**0.6491**	**0.6389**
RMSprop Sigmoid 1 variable Batch 64 3 layers 45 epochs	**0.3017**	**0.3074**
Adam Relu 2 variables Batch 64 3 layers 45 epochs	**0.2180**	**0.2412**
Adam Sigmoid 2 variables Batch 64 3 layers 45 epochs	**0.3033**	**0.3074**
SGD Relu 2 variables Batch 64 3 layers	**0.4473**	**0.4298**
RMSprop Relu	**0.6497**	**0.6389**

2 variables Batch 64 3 layers		
RMSprop Sigmoid 2 variables Batch 64 3 layers	**0.3017**	**0.3074**
Adam Relu 3 variables Batch 64 3 layers 45 epochs	**0.2180**	**0.2412**
Adam Sigmoid 3 variables Batch 64 3 layers 45 epochs	**0.3041**	**0.3074**
RMSprop Relu 3 variables Batch 64 3 layers 45 epochs	**0.6497**	**0.6389**
RMSprop Sigmoid 3 variables Batch 64 3 layers	**0.3017**	**0.3074**