Heidelberg University
Institute of Computer Science

Project report for the lecture Advanced Machine Learning

Prediction of the next SARS-CoV-2 variants

https://github.com/nilskre/AML-covid-project

Team Member: Felix Hausberger, 3661293,
Applied Computer Science
eb260@stud.uni-heidelberg.de

Team Member: Nils Krehl, 3664130,
Applied Computer Science
pu268@stud.uni-heidelberg.de

Plagiarism statement

We certify that this report is our own work, based on our personal study and/or research and that we have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication.

We also certify that this report has not previously been submitted for assessment in any other unit, except where specific permission has been granted from all unit coordinators involved, or at any other time in this unit, and that we have not copied in part or whole or otherwise plagiarized the work of other students and/or persons.

Member contributions

Nils Krehl

Felix Hausberger

Contents

List of Abbreviations

GAN	Generative Adversarial Network
LSTM	Long Short-Term Memory
RNA	Ribonucleic Acid
RNN	Recurrent Neural Network

0 Project Setup

For a detailed description of how to set up the project, please have a look at `https://github.com/nilskre/bomberman_rl/blob/master/README.md`.

1 Introduction

2 Fundamentals and Related Work

2.1 From Probabilistic Language Models to modeling Evolution Theory

A lprobabilistic anguage model tries to approximate the probability distribution

$$P(w_1, ..., w_n) = \Pi_{t=1}^{n} P(w_t | w_1, ..., w_{t-1}) \tag{1}$$

with w_t being a word at position (timestamp) t in a sentence of length n. To build language models Recurrent Neural Networks (RNNs) were used to model such probability distributions.

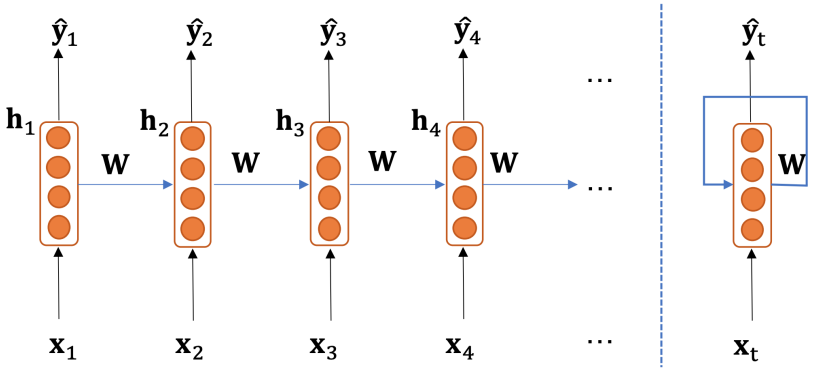

Figure 1: Architecture of a conventional RNN model [1]

At each time step t it outputs a probability distribution $P(w_t | w_1, ..., w_{t-1})$ given the words read so far in the current instance (see Figure 1). Words are read as a vectorized numerical representation, often so-called word embeddings x_t. We then calculate the hidden state h_t by

$$h_t = f(W_h h_{t-1} + W_x x_t + b_1) \tag{2}$$

and the corresponding output porbability distribution by

$$\hat{y}_t = softmax(U_h h_t + b_2). \tag{3}$$

The applied weight matrix is always the same for each time step t giving the RNN its name. One can therefore simplify the unrolled RNN architecture on the left side of Figure 1 to the one on the right, where the hidden state is continuously passed as an input to the next time step. To achieve a better convergence behavior during training, one can also provide the expected

hidden state of time step $t-1$ instead of using the predicted one, which is called teacher forcing. [1]

RNNs are therefore able to process input of arbitrary length and are capable to use information from previous time steps. Unfortunately, they are vulerable to vanishing and exploding gradient problems. The Long Short-Term Memory (LSTM) is a special RNN architecture that solves such vulnerabilities by owning a separate long-term cell state besides a short-term hidden state and is introduces in subsection 2.5. It is able to preserve information over many time steps. [1]

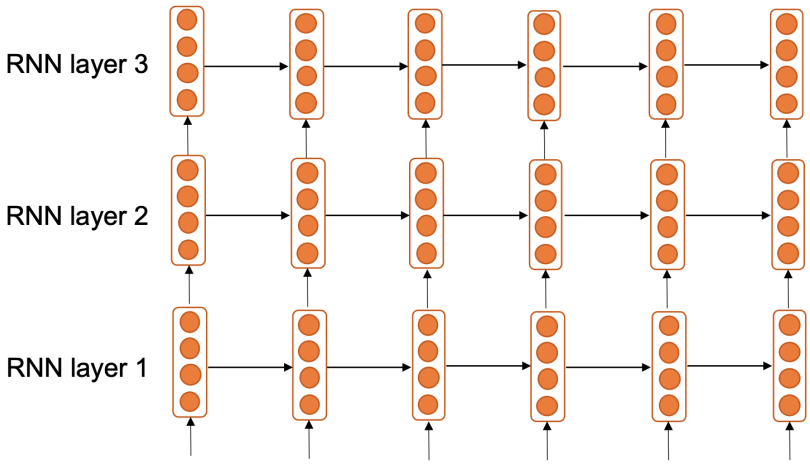

Figure 2: Architecture of a multi-layer RNN [1]

One can also use two RNNs, one passing a sentence from left to right and another one vice versa, with two different weight matrices to model the probability distribution. One therefore simply concatenates the hidden states of each RNN before applying the the weight matrix U and the $softmax()$ function. Also multi-layer RNN can be utilized to generate higher-order features (hidden states) for the prediction task (see Figure 2).

The RNNs architectures used for probabilistic language modeling can be reused in a more complex domain called sequence to sequence modeling for neural machine translation from one lneaguage to another. Here one first tries to learn a fixed-size input representation from an input sequence using an encoder architecture based on an RNN. The so-called context vector is then decoded into a new sequence of words preserving the grammar but owning a different meaning. Sequence to sequence models are introduced together with the LSTM architecture in subsection 2.5. Here the connection to evolution theory can be drawn. Ribonucleic Acid (RNA) sequences made

of a concatenation of nucleotides [1] can be represented textually using the FASTA format. A sequence to sequence model can also be applied in this domain to model how RNA-based viruses change their structure to avoid the detection by the human immune system but still to preserve their infectivity and evolutionary fitness.

We will also look in-depth into the preferable Transformer architecture over sequence to sequence models (see subsection 2.7).

2.2 GISAID EpiFlu

2.3 Domain-Specific Methodologies to create Evolutionary Datasets for Mutation Prediction

2.4 Previous work on Mutation Prediction

Even before the rise of Covid-19 there had been studies trying to predict mutations of RNA viruses. In the collection of [7, 6, 8] the authors predict the mutation positions in hemagglutinins from influenza A virus using logistic regression and plain neural networks and then use the resulting amino acid mutating probabilities to derive possible mutated amnio acids. The same approach was further used for H5N1 neuraminidase proteins.

[4] proved that nucleotides in an RNA sequence can change based on their local neighborhood. Neural networks are used to predict new strains of the Newcastle virus and subsequently a rough set theory based algorithm is introduced to extract the according point mutation patterns.

[3] uses a more modern seq2seq LSTM neural network approach to learn nucleotide mutations between time-series species of H1N1 Influenza virus and the Newcastle virus as mutations can also be influenced by long-distance relations of amino acids. Therefore one hot-encoded RNA sequences of a parent generation preprocessed to words is given as input and the output is the predicted offspring generation evaluated by accuracy to the true offspring generation. The achieved accuracy in this paper is questionably high with 98.9% on the H1N1 Influenza virus and 96.9% on the Newcastle virus, possibly because of overfitting to the few 4.609 samples for H1N1 Influenza virus and only 83 for the Newcastle virus. Our approach therefore tries to increase the number of samples available for training when building the dataset.

Our approach will neither use any of the just mentioned architectures, but uses a Transformer based architecture coupled with a GAN-style training

[1] We restrict the representation of nucleotides solely to their nucleobases parts consisting of the distinct nucleobases guanine, adenine, cytosine and thymine. We therefore do not look at the phosphate group and the five-carbon sugar part.

architecture. Nevertheless we would like to give a short introduction into sequence to sequence models and the underlying long short-term memory components to better point out our architectural decisions .

2.5 Sequence2Sequence Models based on Long Short-Term Memory

The original LSTM unit was introduced in [2] and can be used for language modeling instead of using plain RNNs to prevent running into vanishing gradient problems [5].

[**bla**] introduced sequence to sequence learning by using one LSTM to learn a large fixed-dimensional vector representation of the input that is provided one timestamp at the time and another LSTM to map the so-called context vector to a corresponding output sequence whose length does not need to match with the length of the input sequence. The output sequence is therefore given by the equation

$$p(y_1, ..., y_{T'}|x_1, ..., x_T) = \Pi_{t=1}^{T'} p(y_t|v, y_1, ..., y_{t-1}) \tag{4}$$

where each $p(y_t|v, y_1, ..., y_{t-1})$ is given by the softmax over all words in the vocabulary. Using an LSTM is prefered over a normal RNN as it is used to capture the long range temporal dependencies of the input data, each LSTM uses four layers with 1000 cells each. One finding was also that reversing the input sequence introduces many short term dependencies making optimization easier. The sequence to sequence model approach was evaluated for neural machine translation and reached a 34.81 BLEU score with an output vocabulary of 80k words (160k words input vocabulary, 1000 dimensional word embeddings, 8000 real numbers to represent a sentence).

—

- LSTM: `https://www.researchgate.net/publication/13853244_Long_Short-term_Memory`
- LSTM: `https://www.isca-speech.org/archive/archive_papers/interspeech_2012/i12_0194.pdf`
- Seq2Seq: `https://arxiv.org/abs/1409.3215`

2.6 Applying Generative Adversarial Networks

- Covid-Paper: `https://arxiv.org/pdf/2008.11790.pdf`

2.7 Transformer and Attention Mechanism

- Improvement: https://arxiv.org/abs/1706.03762

2.8 Other Techniques

- NNs/SVMs: https://bsb-eurasipjournals.springeropen.com/articles/10.1186/s13637-016-0042-0
- BiLSTM: https://science.sciencemag.org/content/371/6526/284

3 Approach

3.1 Dataset Creation

3.2 Data Preprocessing

- DNA Sequencing
- DNA Sequence Tokenization for Amino Acid Dictionary
- DNA Sequence Padding

3.3 Model Architecture

3.4 Training Process

4 Experimental results

5 Conclusion

References

[1] Michael Gertz. "Text Analytics - Text Classification". 2020. (Visited on 08/15/2021).

[2] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *ResearchGate* (1997). URL: https://www.researchgate.net/publication/13853244_Long_Short-term_Memory (visited on 08/13/2021).

[3] Takwa Mohamed et al. "Long Short-Term Memory Neural Networks for RNA Viruses Mutations Prediction". In: *Hindawi* (2021). URL: https://www.hindawi.com/journals/mpe/2021/9980347/ (visited on 08/11/2021).

[4] Mostafa Salama, Aboul Ella Hassanien, and Ahmad Mostafa. "The prediction of virus mutation using neural networks and rough set techniques". In: *EURASIP Journal on Bioinformatics and Systems Biology* (2016). URL: https://bsb-eurasipjournals.springeropen.com/articles/10.1186/s13637-016-0042-0 (visited on 07/05/2021).

[5] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. "LSTM Neural Networks for Language Modeling". In: *ISCA Archive* (2012). URL: https://www.isca-speech.org/archive/archive_papers/interspeech_2012/i12_0194.pdf (visited on 08/12/2021).

[6] Guang Wu and Shaomin Yan. "Prediction of mutations engineered by randomness in H5N1 hemagglutinins of influenza A virus". In: *SpringerLink* (2007). URL: https://link.springer.com/article/10.1007/s00726-007-0602-4 (visited on 08/13/2021).

[7] Guang Wu and Shaomin Yan. "Prediction of mutations engineered by randomness in H5N1 neuraminidases from influenza A virus". In: *SpringerLink* (2007). URL: https://link.springer.com/article/10.1007/s00726-007-0579-z (visited on 08/13/2021).

[8] Guang Wu and Shaomin Yan. "Prediction of mutations in H1 neuraminidases from North America influenza A virus engineered by internal randomness". In: (2008). URL: https://link.springer.com/article/10.1007/s11030-008-9067-y (visited on 08/13/2021).