COVID-19 and International Measures

Dahlia Shehata*, Vaakesan Sundrelingam**, and Shiyu He**

*David R. Cheriton School of Computer Science
**Department of Statistics and Actuarial Science
University of Waterloo
{d3shehat, v2sundrelingam, s35he}@uwaterloo.ca

Abstract

COVID-19 breakthrough has affected the life of millions of people. In order to reduce the number of infected cases and alleviate the burden on healthcare systems, strict measures were imposed in each country including lockdowns, quarantines, curfews, border closures and travel restrictions. This project uses a novel dataset which is a combination of two main sets: COVID-19 Government Measures Dataset and an up-to-date version of Kaggle's Novel CoronaVirus 2019 Dataset to investigate the effect of government measures on the number of confirmed COVID-19 cases worldwide. We propose variants of sequence-to-sequence models with both LSTM and GRU units equipped with different types of BERT embeddings including vanilla BERT and BioBERT to capture the overall trend per country. We also explore different joining techniques for these embeddings such as truncated concatenation, mean aggregation or using an autoencoder. Our best model outperforms the naive LSTM baseline by a significant factor and with a prediction ability covering a wide time range. To the end, we perform a comparative study between the different experimented models with a reflection on limitations, research gaps and open problems.

Index Terms: Deep Learning, Seq2seq, NLP, COVID-19.

1 Introduction

The COVID-19 pandemic has profoundly affected the world since its first outbreak in December 2019. As of 19th April 2021, there have been over 141 million confirmed cases and 3.01 million deaths globally [1]. Consequently, research effort has been oriented towards leveraging state-of-art deep learning models for predicting case counts. For example, Arora et al. [2] use Long Short-Term Memory (LSTM) models [3] to forecast weekly COVID-19 cases of the Indian states with an error rate of 3%. Saba and Elsheikh [4] employ Auto Regressive Integrated Moving Average (ARIMA) and neural networks based on approximate reasoning architecture (NARANN) models to forecast the daily Covid-19 cases of Egypt with a prediction error of 5%. Shastri et al. [5] propose recurrent neural networks (RNN) variants to predict monthly cases of both India and US with a minimum error rate of 2.0%. Abdulmajeed et al. [6] develop an ensemble model that combines ARIMA, additive regression model and Holt-Winter Exponential smoothing method offering a real-time update of COVID-19 case predictions of Nigeria. As it was shown, previous literature is limited by two constraints: country-specific cases and a narrow prediction time frame. Such drawbacks are overcome by our proposed model. In another context, governments are taking continuous measures to limit the human and economic impact of COVID-19. Although, this factor contributes to the number of COVID-19 cases, it is overlooked in prior works. In this paper, we exploit both the number of confirmed cases globally along with the imposed measures to build a *sequence-to-sequence* model that is able to predict the number of confirmed cases for each country over longer periods with a mean square error (MSE) of 0.04.

Our contributions can be summarized as follows: 1) A new clean dataset incorporating up-to-date numbers of confirmed, death and recovered cases; governmental measures; and daily, weekly and yearly correlation features. 2) Three types of joining techniques of BERT [7] and BioBERT [8] measure embeddings passed to the model which are truncated concatenation, mean aggregation and using an autoencoder. 3) Eight variants of seq2seq model including combinations of GRU/LSTM, Unidirectional/Bidirectional and with/without attention. 4) A detailed comparison between the performance of seq2seq variants and also with the LSTM baseline. 5) The code and results are available at: https://github.com/Dahlia-Chehata/COVID-19-measures

Course project in *Topics in Statistics: Deep Learning* (STAT946, 2021 Winter).

The rest of the paper is organized as follows. Section 2 reviews the background and related works. Section 3 illustrates our proposed methods in terms of dataset, preprocessing and modeling. The details of the conducted experiments are described in Section 4. Section 5 summarizes the main results. Limitations and imminent future research work are discussed in Section 6. Finally, we conclude the discussion in Section 7.

2 Background and Related Works

Previous literature using government measures to predict the number of COVID-19 cases are rare; and they mostly rely on mathematical, statistical and epidemiological models [9], [10], [11]. Wang et al. [9] propose a survival-convolution model for predicting key statistics of COVID-19 daily cases while comparing the effectiveness of mitigation measures across countries. Liu et al. [10] combine COVID-19 case data with mobility data to estimate a modified susceptible-infected-recovered (SIR) model in the United States. Pedersen and Meneghini [11] propose to model COVID-19 dynamics with a SIQR (susceptible – infectious – quarantined– recovered) model rather than SIR to quantify the effect of the restrictions imposed in Italy. To the best of our knowledge, we are the first to leverage deep learning and natural language processing (NLP) methods to identify the effect of government measures on the predicted COVID-19 cases.

3 Methods

Our methods address three main aspects: dataset preparation, preprocessing and modelling.

3.1 Dataset

We use a novel dataset which is a combination of two other ones:

- **COVID-19 Government Measures Dataset:** is published by ACAPS reports covering the regulatory measures passed by 183 countries to combat COVID-19 and reported as text. These measures are grouped into five categories: social distancing, movement restrictions, public health measures, social and economic measures, and lockdowns. The data is retrieved from https://www.acaps.org/covid-19-government-measures-dataset.
- **Kaggle's Novel CoronaVirus 2019 Dataset:** collected originally by the John Hopkins University Center for Systems Science and Engineering [12]. The raw data in this version between 2020-01-21 and 2021-02-26 was aggregated and curated by user SRK on Kaggle. It can be retrieved from https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset

To prepare the data for our model, operations such as augmentation, cleaning and feature engineering are performed.

3.1.1 Augmentation

In order to calculate the yearly autocorrelation feature (as discussed in 3.1.3), we needed to augment the data to cover at least a whole year. As a result, the limited version of the second dataset was not enough. An up-to-date version was scraped directly from the GitHub repository https://github.com/CSSEGISandData/COVID-19 to cover 446 days between 2020-01-22 and 2021-04-11, aggregated and cleaned for the models.

3.1.2 Cleaning

After data augmentation, the datasets are combined in order to associate the government regulations and the number of confirmed COVID-19 cases reported for each country. As COVID-19 death cases are reported by Province/State, we aggregate them at the country level to match the granularity of COVID-19 Government Measures dataset. The datasets are then joined by *"Country"* and *"ObservationDate"*. The countries and dates which did not belong to both datasets are dropped during the inner join. In addition, existing inconsistencies between country names across the two datasets are resolved. Duplicate variables {"ObservationDate", "DATE_IMPLEMENTED"} and variables irrelevant to the problem {*"ID"*, *"ISO"*, *"Pcode"*, *"ADMIN_LEVEL_NAME"*, *"LOG_TYPE"*, *"TARGETED_POP_GROUP"*, *"NON_COMPLIANCE"*, *"SOURCE"*, *"SOURCE_TYPE"*, *"LINK"*, *"Alternative source"*} are eliminated. It is important to mention that some assumptions are made in order to combine the two datasets. For example, if no regulation is passed on a given day for a given country, the previous measure is carried forward. In addition, if multiple measures are issued on the same day, the measures are concatenated into a single string. The limitations of this approach and our attempts to resolve them are discussed in 3.2.1.

3.1.3 Feature Engineering

The dataset is treated as a time series since there is a dependence between the current observation and observations at prior time points. It would be simply naive to rely entirely on the measures for an accurate prediction while neglecting the time factor. We observe that the autocorrelation is at its highest degree for adjacent days, and gradually decreases with more time lags. An RNN, and especially LSTM-based, model is considered appropriate for modeling time series data, as it can pick up lags of unknown correlations in each country. We also know that there is a cyclical behaviour in the number of COVID-19 cases across seasons. As a result, we decide to augment the data with features capturing the daily and monthly periodicity. One naive approach is to use one-hot or ordinal encoding of days and months; but the latter does not capture the cyclical relationship of the variable levels (e.g. December and January are as "close" as November and December). Therefore, we engineer sine and cosine transformations of the *day of the week*, day and month to be used as input features to the model in addition to the original data. Moreover, we added a "yearly autocorrelation" feature to capture the correlation between the number of cases in the current year and the prior one. In order to achieve that, countries with fewer than 366 observations are dropped. A *"year_mod"* feature is also added to denote the fraction of the current number of years elapsed over the total number of years elapsed for the time series of a given country. As a final step, we center and scale all the variables for faster convergence [13].

3.2 Preprocessing

3.2.1 Embeddings

The problem with BERT embeddings mechanism described in A.1 is that it assumes the existence of some dependency between the words in a sentence. However, this is not always the case for our problem. In the case where multiple measures are passed in a single day, multiple sentences are concatenated together with a dot. In our baseline embeddings, we truncate the concatenated sentences to 100 tokens and use **BERT-as-Service** [14] to quickly generate embeddings for the measures with the *"bert-base-uncased"* model from the *"transformers"* library. We also compare the performance with the **BioBERT** model which was trained on a corpus of PubMed abstracts with a vocabulary of 4.5B words. The motivation behind this choice is to employ a vocabulary of tokens more relevant to the vocabulary of COVID-19 measures. To resolve the issue of sentence independence in the embeddings, we attempted two variations.

- **Mean Aggregation:** where a BERT embedding vector is generated for each measure and then the mean is computed across the number of measures generating a single vector.
- **Autoencoded Dimensionality Reduction:** where a stacked list of BERT embeddings for the different measures at a single time are fed to the encoder part of the autoencoder. The output of the encoder is used as final "paragraph" embeddings. Further details are described in A.1.

3.2.2 Sequence Generation

In order to generate the sequences, we use the features produced in 3.1.3 which include: 1) sine and cosine transformations of the day of the week, day of the year, month 2) yearly autocorrelation is added as an engineered date feature. 3) *"year_mod"* feature. In addition, we introduce a yearly lag for the number of confirmed cases which is computed with the help of the autocorrelation function (ACF) of 366 days (year) lag using the following formula:

$$\text{yearly\_lag} = (0.5 * ACF[365]) + (0.25 * ACF[364]) + (0.25 * ACF[366])$$

BERT embeddings are also added as a list of 768 new features. In order to improve the accuracy, the number of death and recovered cases are appended to the list [1]. The aforementioned data represent the numerical features. As categorical features, the *"country"* feature is added, in a later step, to distinguish between time series behaviour across the countries using one-hot-encoding. All of these features are concatenated generating a running sequence of features over time. Our proposed model takes a sequence as input over a pre-specified time window and outputs a sequence too. Our choice for the input and output time windows are 30 and 15 respectively. The reason behind such choice is: 1) RAM limitations that prevent taking longer sequences as input; and the bad performance of the shorter ones 2) The output window value is selected in consideration of the 14-day incubation period of COVID-19 [15]. So we expect that on the 15th day, the person is either infected or not.

[1] One may argue that the number of death and recovered cases are not relevant as inputs. Actually, they are not controlled variables such as the regulations and they may have an undesirable effect on the overall prediction ability. However, we reject this hypothesis since the number of death and recovered cases are causally related to government measures. They have an implicit impact on the imposition of new measures or the revocation of existing regulations. Both features are centered and scaled before adding to the features' list.

3.3 Model

3.3.1 Baseline

We first implement an LSTM baseline to predict the number of COVID-19 confirmed cases for all countries. The reason behind this choice is that we believe that the general trend can be captured using the baseline's memory unit. The model simply takes as input the number of confirmed cases for all countries and outputs equivalent predictions. It is trained with 32 hidden states to learn and weigh the features at different time lags for each country with an input window of 30 days and an output window of 15 days. The major drawback of the LSTM is that it does not take into consideration government measures imposed to limit the pandemic propagation (which is the core purpose of our project). As a result, it is unable to pick up the individual patterns for each country with different speeds of growth resulting from differences in population and policies (refer to A.2). We just wanted to quantify the effect of government measures on the prediction ability compared to a trivial LSTM that relies only on the time component in data. The result of the training MSE for the baseline is 0.0373 and the testing MSE is 0.1665.

3.3.2 Seq2seq

The main model we adopted is *sequence-to-sequence*. The Encoder-Decoder architecture is used to model time prediction problems in general which aligns with our objective. The goal is to provide a multi-step forecast of $x_{t+1}, \dots x_{t+q}$ based on historical data $x_{t-p}, \dots, x_t$. To give an abstract representation of the input and output of the overall architecture. Suppose the encoder takes an input sequence x over a time period p and generates an output sequence y over a time period q. Each sequence x includes time-dependent features such as those extracted in 3.1.3 and can be represented as $x_0, \dots, x_{f-1} \in R^f$; a single categorical variable with $i - f$ different values (for simplicity), i.e. the one hot encoding vector of countries with values $x_f, \dots, x_{i-1} \in R^i$; static numerical data, i.e the yearly autocorrelation, that are represented as $x_i, \dots, x_{j-1} \in R^j$ and measure embeddings denoted as $x_j, \dots, x_{k-1} \in R^k$. We can represent the input and output vectors of seq2seq as:

$$\begin{bmatrix} x_{0,t-p} & x_{1,t-p} & \dots & x_{f-1,t-p} & \dots & x_{j-1,t-p} & \dots & x_{k-1,t-p} \\ x_{0,t-p+1} & x_{1,t-p+1} & \dots & x_{f-1,t-p+1} & \dots & x_{j-1,t-p+1} & \dots & x_{k-1,t-p+1} \\ \vdots & & \dots & \vdots & & & & \\ x_{0,t} & x_{1,t} & \dots & x_{f-1,t} & \dots & x_{j-1,t} & \dots & x_{k-1,t} \end{bmatrix} \longrightarrow \begin{bmatrix} y_t \\ y_{t+1} \\ \vdots \\ y_{t+q} \end{bmatrix}$$

where $f =$ number of time-dependent data, $i - f =$ length of the one-hot-encoder vector of the categorical variable, $j - i =$ number of static numerical data, and $k - j =$ length of the embedding vector. In our case, $p = 30$, $q = 15$, $f = 10$, $i - f = 177$, $j - i = 1$ and $k - j = 768$. These dimensions are thoroughly explained in A.3.

- **Encoder:** We applied 4 encoder variants with combinations of (LSTM, GRU) and (Unidirectional, Bidirectional). The LSTM-based model gave the best performance, that is why we provide a description about its mechanism:

  1) The **Unidirectional** encoder is composed of a series of LSTM cells, composed of three gates: input gate i_t, forget gate f_t, output gate o_t. Each cell takes as input the previous hidden layer h_{t-1}, the previous cell state c_{t-1}, and the input features from the current time step x_t; and outputs a cell state c_t and a hidden state h_t [3]. For any input time series at time $t - p, \dots t$, the three gates control the flow of information through sigmoid layers:

  $$\begin{aligned} f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f), \\ i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i), \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o), \end{aligned}$$

  where W and U are weight matrices and b is the bias vector, all of which can be learned during training. The hidden state h_t is also generated through a *tanh* layer and passed onto the next cell:

  $$\begin{aligned} \tilde{c}_t &= \sigma(W_c x_t + U_c h_{t-1} + b_c) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\ h_t &= o_t \circ tanh(c_t) \end{aligned}$$

  After time step $t - p, \dots, p$, the hidden state of h_t becomes the input to the decoder.

  2) The **Bidirectional** LSTM has two interconnected hidden layers which process the time series data in two directions at the same time [16]. Besides the gates and cells set up as above, we also have equivalent cells in the opposite direction $\overleftarrow{f}_t, \overleftarrow{i}_t, \overleftarrow{o}_t, \overleftarrow{c}_t, \overleftarrow{h}_t$, and the final hidden states are computed as: $h_t = \overrightarrow{h}_t \oplus \overleftarrow{h}_t$

- **Decoder:** We experimented with 4 decoder variants with combinations of (LSTM, GRU) and (without Attention, with Attention). We limit our theoretical explanation to the LSTM as in the encoder case.

  1) The decoder **without Attention** uses the last hidden state of the encoder, concatenated with the time independent features as input. The Decoder part is built by a series of LSTM cells, so the forecast obtained from one LSTM cell is passed to the next one. The output of the hidden state of each LSTM cell $h_{t+1}, \dots h_{t+q}$ is fed into a fully connected layer $y_t = \sum_j w_j h_j$ and a dropout layer. Finally, it provides the prediction of $y_{t+1}, \dots y_{t+q}$, which is the output sequence.

  2) With the **Attention mechanism**, the input can be recursively processed while preserving its internal hidden state. At each time step t, the LSTM reads x_t and updates its hidden state h_t with self attention:

$$s_{t,i} = q_i^T h_t,$$
$$a_{t,i} = \frac{e^{s_{i,t}}}{\sum_i e^{s_{i,t}}}$$
$$h'_t = \sum_i a_{t,i} h_t$$

  Finally, each updated hidden state h'_t is fed into a fully connected layer and a dropout layer, and this provides the forecast of $y_{t+1}, \dots y_{t+q}$.

Figure 1 illustrates the architecture of our proposed model that combines mean aggregated or autoencoded embeddings with time-series features as input to the LSTM-based or GRU-based seq2seq architecture.

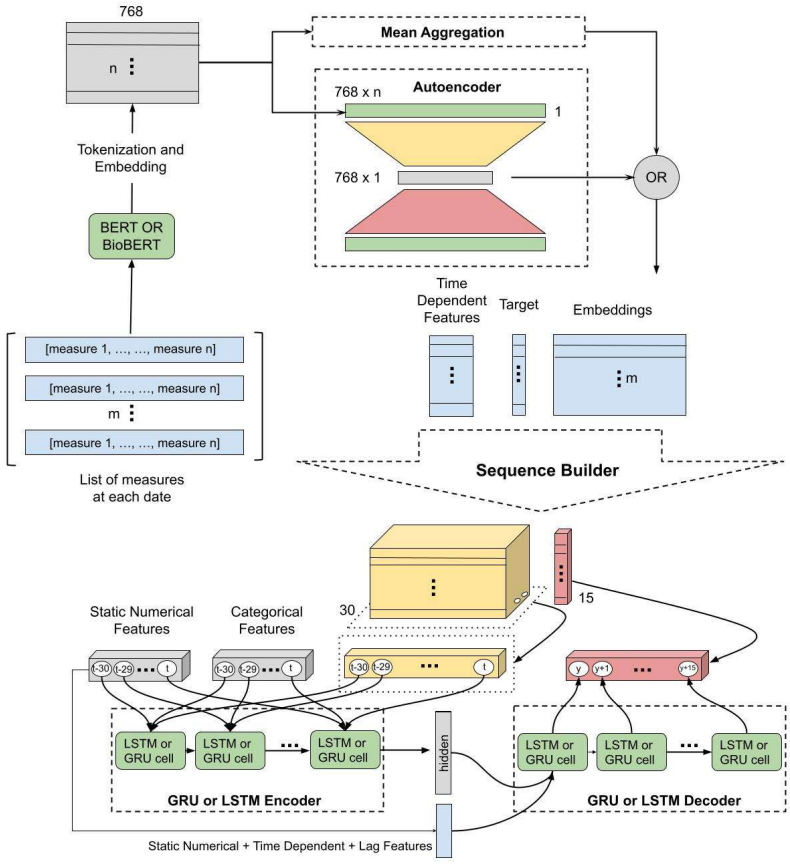


Figure 1: Model Architecture: the embeddings are passed to BERT/BioBERT to be encoded independently, then either mean aggregation or autoencoder is employed to join the multiple embeddings issued on the same date. The latter along with time dependent features and the target are passed to the sequence builder that generates an input and output sequences for the model.

4 Experiments

We split the dataset into training and testing sets with a ratio of 9:1 which is equivalent to the split date: 2021-02-22. The sizes of both training and testing sets are 59368 and 6272 respectively. Since it is important to capture the time dependencies, we refrained from shuffling the training data.

4.1 Training Settings

We tried different values of epochs such as 6, 12, 24; but the latter gave the best results. We tried changing the number of steps for each epoch by multiplying the original value *trainset_len* with factors of 2 and 3. Slight improvement was detected. We changed the number of RNN layers with values 1, 2, 4 and 8, but no major change was observed. In addition, no concrete difference was found between values of 100 and 200 for the hidden layers. We selected the best fine-tuning learning rate among (1e-5, 1e-3, 1e-2, 3e-5, 3e-3, 3e-2) which is 1e-3 as shown in Figure 5. We used a batch size of 128, a dropout of 0.2, a *weight_decay* of 1e-2. Due to time and resource scheduling constraints, we have not tried changing these values. The choice was based on domain knowledge from previous similar experiments. We used MSE loss function, Adam optimizer and *"OneCycleL"* scheduler [17].

4.2 Experiment Resources

Overall, we perform a total of 33 experiments (excluding those used for parameter tuning in 4.1): 32 for seq2seq variants and 1 for the baseline. In terms of experiment resources, the longer the window size of the sequences, the more resources were required. Our experiments were conducted on:

- **Colab Pro** with a GPU **Tesla P100-PCIE-16GB** and 25GB of RAM for data preparation, training and testing.
- **Compute Canada** with 2 GPUs **V100-SXM2-32GB** and 64GB of RAM for sequence generation.

5 Results

Table 1 summarizes our results on the testing dataset for all experimented Seq2seq variants with different embeddings in terms of mean squared error (MSE). The best MSE is **0.04** which is obtained from an LSTM-based Seq2seq using mean aggregated BERT embeddings. Detailed result analysis is provided in A.4.

Model Type/ Embedding Type			**Truncated Concatenation**		**Mean Aggregation**	**Autoencoded**
			BERT	BioBERT	BERT	BERT
GRU-based	Unidirectional	no Attention	0.11058	0.17138	0.07801	0.06037
		Attention	0.18757	0.11904	0.07801	0.08581
	Bidirectional	no Attention	0.10027	0.09600	0.04720	0.05273
		Attention	0.10493	0.09656	0.07294	0.07342
LSTM-based	Unidirectional	no Attention	0.10111	0.08803	**0.04143**	0.05389
		Attention	0.21591	0.20762	0.06425	0.07420
	Bidirectional	no Attention	0.08244	0.11229	0.04378	0.05538
		Attention	0.20796	0.10198	0.06493	0.06848

Table 1: MSE values for the Seq2seq variants with different embedding types and different embedding joining schemes.

Figure 2 shows our best model's sequence prediction for the next 15 days depending on input features from a prior month.

6 Discussion and Future Work

Our model equipped with mean aggregated BERT embeddings introduces over 75% improvement over an LSTM baseline. The model determines the spread of the virus (represented by the predicted number of confirmed cases) with a high degree of accuracy, using both controlled variables such as government measures and uncontrolled ones such as the number of COVID-19 deaths and recoveries without the high cost of widespread testing.

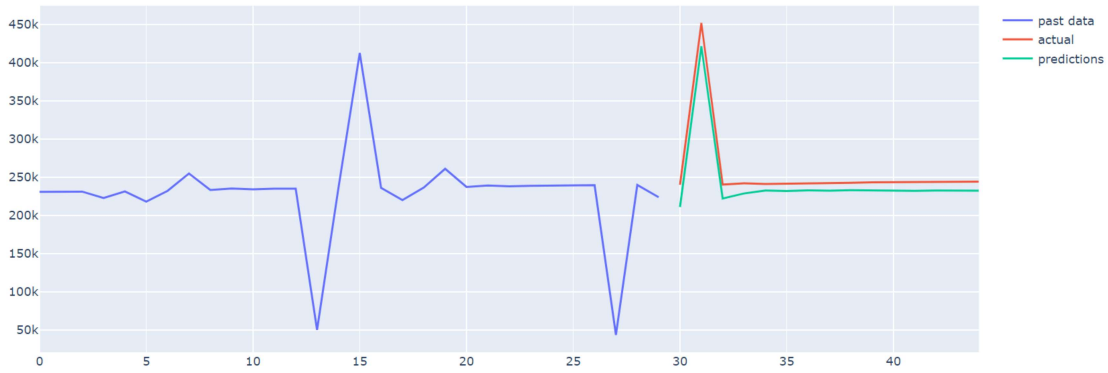


Figure 2: Sequence predictions of the best model-embedding combination on the testing set. The blue line is a the input sequence of features covering a time frame 30 days. The red line is the original sequence values for the next 15 days. The green line represents the best model's predictions.

Nonetheless, there is still room for improving the current implementation and covering research gaps. For example, the model can take additional controlled variables into account such as demographics and measures of policy adherence or policy enforcement. As it was previously described, our model computes the yearly autocorrelation, using the 2020-2021 year to be more specific, which was proved to be very efficient in terms of prediction accuracy. However, new factors have recently emerged and they need to be taken into account. Vaccine distribution rates, vaccine types, the number of vaccinated individuals and even new variants of the virus are all factors that did not exist in the past year, and their effects are still to be observed, measured and then predicted.

Another shortcoming of our current model is that the generated embeddings are fixed, and that is because the embedding generation and the seq2seq model are completely independent from each other. Each part acts as a standalone model. As a result, the pre-trained weights of the transformer networks of both BERT and BioBERT embeddings are used to generate fixed embeddings that are passed to the seq2seq model. In the backward pass of the seq2seq, the changes cannot be propagated to the embeddings' transformer, hence its weights are not updated. It is worth mentioning that we tried to overcome this drawback by implementing an end-to-end transformer network which would generate a 1x768 dimensional embedding for the measures, and in which the weights of the transformer could be tuned simultaneously with the seq2seq model. However, we were not successful due resource limitations. Adding BERT and BioBERT transformers as trainable layers in the *sequence-to-sequence* model itself would delegate the sequence generation to the model which was a big limiting factor for us. This idea is worth exploration though in a related future research work.

In addition to these end-to-end variations, BioBERT proved its outperformance over vanilla BERT using the same truncated embedding joining scheme. It was also proved that both Mean Aggregation and Autoencoded Dimensionality Reduction are better than measures' concatenation. As a result, it is important to try BioBERT with these techniques. We believe that this will give the best performance of all experimented approaches since BioBERT is more adequate to the COVID-19 jargon.

We also observed that our model was sensitive to the size of the input time window when generating the feature sequences. We tried an input window size of 10 and it gave poor results compared to our final input window size which is 30. Sequences covering longer time frames may even produce better results. One other idea that we were not able to explore due to RAM constraints. The same can be said about the embeddings' size that was limited to 768 in our case.

As an extension to the current work, it would be equally interesting to predict the government measure(s) issued given the target numbers of confirmed, death and recovered cases.

7 Conclusion

To conclude, we have presented a reliable and efficient technique for predicting the number of COVID-19 confirmed cases using government measures; number of deaths and recoveries; and time-series features captured across the last year. By mean aggregating or reducing the dimensionality of BERT (or its variants) embeddings of the measures issued on the same day, we were able to forward them to a seq2seq model. The MSE of the generated predictions was reduced by a significant factor when compared to the output of an LSTM baseline. The problem is new and was not addressed previously by the deep learning research community. We shared our experiments' details including code and training settings. We also covered the limitations and research gaps of our work; and proposed new research ideas for future work that may contribute to healthcare advancement.

Acknowledgement

We would like to thank Prof. Ghodsi, Aref and Mojtaba for the their instructive comments and guidance during the course.

References

[1] WHO. *Coronavirus disease (COVID-19) Situation dashboard*. Last checked on Apr 19, 2021. 2020.

[2] P. Arora, H. Kumar, and B. K. Panigrahi. "Prediction and analysis of COVID-19 positive cases using deep learning models: A descriptive case study of India." *Chaos, Solitons and Fractals*, vol. 139 (2020).

[3] S. Hochreiter and J. Schmidhuber. "Long short-term memory". *Neural computation*, vol. 9 (1997).

[4] A. I. Saba and A. H. Elsheikh. "Forecasting the prevalence of COVID-19 outbreak in Egypt using nonlinear autoregressive artificial neural networks." *Process safety and environmental protection*, vol. 141 (2020).

[5] S. Shastri, K. Singh, S. Kumar, P. Kour, and V. Mansotra. "Time series forecasting of Covid-19 using deep learning models: India-USA comparative case study". *Chaos, Solitons and Fractals*, vol. 140 (2020).

[6] K. Abdulmajeed, M. Adeleke, and L. Popoola. "Online forecasting of COVID-19 cases in Nigeria using limited data". *Data in brief*, vol. 30 (2020).

[7] J. Devlin, M. Chang, K. Lee, and K. Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019.

[8] J. Lee et al. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". *Bioinformatics* (2019).

[9] Q. Wang, S. Xie, Y. Wang, and D. Zeng. "Survival-Convolution Models for Predicting COVID-19 Cases and Assessing Effects of Mitigation Strategies". *Frontiers in Public Health*, vol. 8 (2020).

[10] M. Liu, R. Thomadsen, and S. Yao. "Forecasting the spread of COVID-19 under different reopening strategies". *Scientific Reports*, vol. 10 (2020).

[11] M. Pedersen and M. Meneghini. "Quantifying undetected COVID-19 cases and effects of containment measures in Italy: Predicting phase 2 dynamics" (2020).

[12] E. Dong, H. Du, and L. Gardner. "An interactive web-based dashboard to track COVID-19 in real time". *The Lancet Infectious Diseases*, vol. 20 (2020).

[13] Y. A LeCun, L. Bottou, G. B. Orr, and K. Muller. "Efficient backprop". In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[14] H. Xiao. *bert-as-service*. https://github.com/hanxiao/bert-as-service. 2018.

[15] S. A. Lauer et al. "The incubation period of coronavirus disease 2019 (COVID-19) from publicly reported confirmed cases: estimation and application." *Annals of internal medicine*, vol. 172 (2020).

[16] M. Schuster and K. K. Paliwal. "Bidirectional recurrent neural networks." *IEEE transactions on Signal Processing*, vol. 45, no. 11 (1997), pp. 2673–2681.

[17] L. N. Smith and N. Topin. *Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates*. 2018. eprint: 1708.07120.

A Appendix

A.1 Embeddings

In order to capture relevant information from the regulations and measures passed, distributed vector representations are used to create embeddings from the natural language text data. The basis for encoding a single sentence was BERT. The tokenization process creates a unique representation of a single sentence (or a pair of sentences - not relevant to our work) and a multi-layer bidirectional transformer encoder creates a unique 1x768 dimensional embedding for that sentence. A key aspect of BERT is the bidirectional self-attention. The masked language model training objective uses the masked word as a query, computes the similarity with the other words in the sentence, and outputs a weighted sum of the values associated with those words. This is how BERT embeddings are able to capture left and right context for every word in the sentence.
The major drawback of this mechanism is that it does not suit our data perfectly, especially for the cases where we have multiple concatenated regulations at a single point of time. The order of the concatenated sentences does not really matter in such cases. As a result, we propose two other joining schemes for the embeddings to remove the dependency of the unrelated measures.

- **Mean Aggregation:** On a day with n measures passed, we can generate n 1x768 dimensional embeddings using BERT. Therefore the measures passed on that day can be encoded as a 1xnx768 dimensional "paragraph" embedding. In order to combine this information with the features engineered earlier, we have to reduce this to a 1x768 dimensional embedding, while retaining information about all the component sentences. The maximum size along the dimension at index 1 (the maximum number of measures passed on a single day) was 29. Therefore we pad the 1xnx768 embeddings to 1x29x768. We take the mean along dimension 1 (the n sentences) and return a 1x768 dimensional embedding as needed. These are then combined with the other features and supplied to the model.
- **Autoencoded Dimensionality Reduction:** We begin with the same 1x29x768 dimensional embedding from before. We pass the 3 dimensional embedding to an autoencoder which consists of a flattening layer, 3 linear layers (the encoder) resulting in an output of size 1x768, and 3 additional linear layers (the decoder) that return the input to its original size. In the autoencoder, the targets are the same as the inputs. We take the output from the encoder as our 1x768 "paragraph" embedding.

To better clarify the difference between the three joining schemes: truncated concatenation, mean aggregation and autoencoded dimensionality reduction, let's use a toy example. Consider the following two measures passed on the same day:

['Wear masks.', 'Stand six feet apart']

With concatenation, we need to combine and truncate these measures to *max_length* tokens: *['wear masks stand six']*. After tokenization, this may become [[6, 1, 4, 2]]; and after embedding with BERT, this may become [[0.5, 0.4, 0.2, 0.8]].

The "paragraph embedding" procedure begins the same for both the mean aggregation and autoencoder embeddings. We begin by tokenizing the individual sentences: [[6, 1], [4, 2, 3, 5]]. Then, the individual sentences are padded to the *max_length*: [[6, 1, 0, 0], [4, 2, 3, 5]] and the embeddings are generated for each sentence independently: [[0.1, 0.5, 0.2, 0.4], [0.7, 0.3, 0.6, 0.9]].

In mean aggregation, the sentences are averaged along the axis corresponding to the number of measures passed for a single day: [[0.4, 0.4, 0.4, 0.65]], in contrast with the autoencoder where this lower dimensional representation is learned instead.

A.2 LSTM Baseline Analysis

Our choice of the LSTM as a baseline relied on the observation of the increasing curve of COVID-19 confirmed cases for almost all countries as shown in Figure 3. Figure 4 illusrates exemplary results of the LSTM baseline for 4 random countries: Afghanistan, Albania, Niger and Mexico. The results prove that despite using the same input/output window sizes as in seq2seq, LSTM is unable to capture individual patterns for each country by simply relying on the past number of confirmed cases.

A.3 Seq2seq in depth

In 3.3.2, we simplified the representation of the input and output vectors of the seq2seq for readability. Originally, the sequence builder generates 2 lists of sequences: X‘ and Y' that are used as part of the input to the model. Each sequence x' in the input list is of dimensions (*input_window_size*, *input_feature_length*) which is (30, 778) in our case. Each sequence y' in the output list is of dimensions (*output_window_size*, *output_feature_length*) which is (15, 779) in our case. The second dimension for both x' and y' represents the number of features gathered during the feature engineering process in 3.1.3, additional time-dependent data such as the number of deaths and recoveries, our target variable which is the number of confirmed cases and

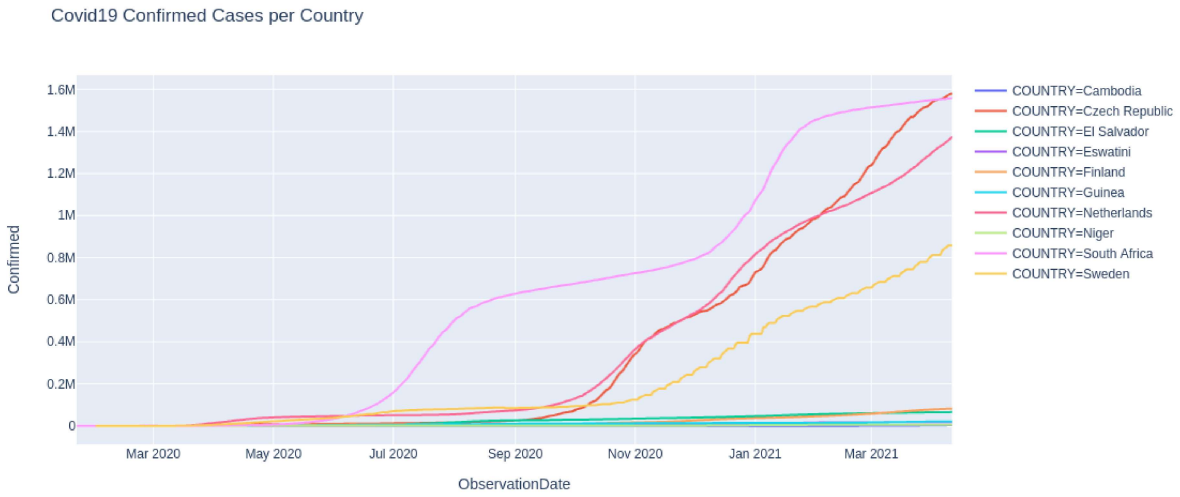
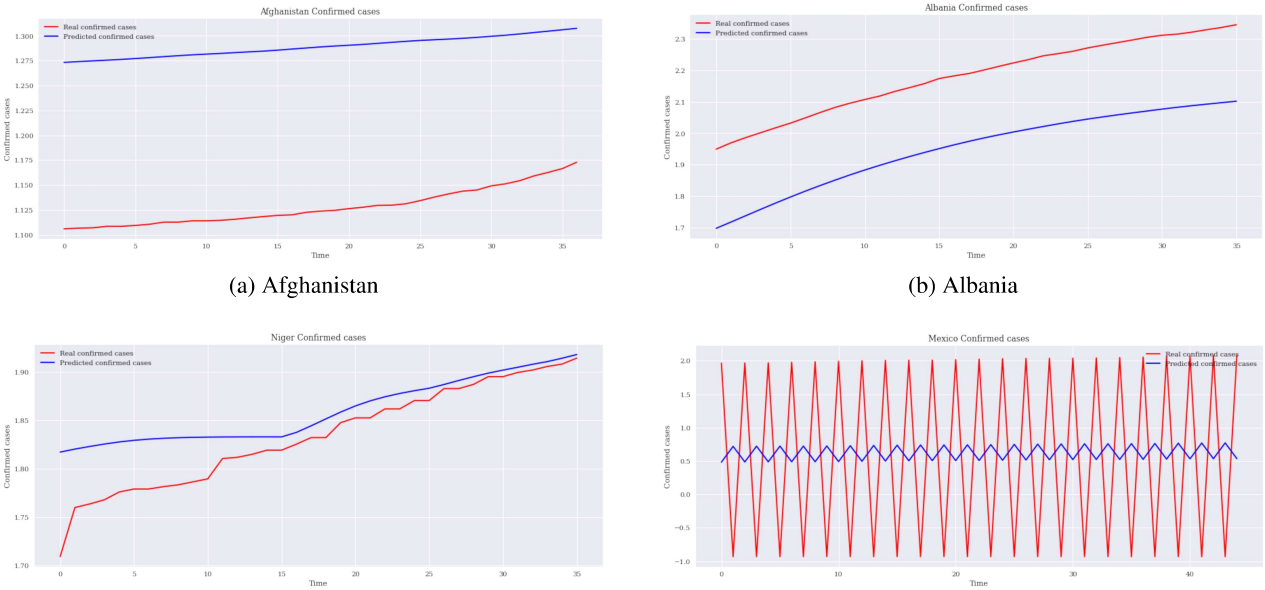


Figure 3: The number of confirmed cases for 10 random countries for the last 466 days.



(a) Afghanistan

(b) Albania

(c) Niger

(d) Mexico

Figure 4: LSTM baseline results for 4 random countries: The red line represents real confirmed cases and the blue line represents the baseline predictions.

the 768 features of the embedded measures. For an input sequence x', these features are: [*confirmed*, *Deaths*, *Recovered*, *dayofweek_sin*, *dayofweek_cos*, *day_sin*, *day_cos*, *month_sin*, *month_cos*, *year_mod*] + 768 embeddings features.

You may have already noticed that the second dimension of the output sequence y' is larger than that of x' with 1 value. This difference is due to the *last_year_lag* value that we introduced in 3.2.2. We only have data that covers approximately one year, that is why the lag is a single value. The rest of the features are the same as explained for x'. However, X' and Y' are not the data that are used as input to the model. The model takes X and Y.

To generate X from X', we concatenate X' which is the time-dependent features of dimensions (*batch_size*, 30, 778) with static numerical features (i.e. yearly autocorrelation in our case) and a single one-hot-encoded categorical variable vector (i.e. countries which were reduced to 177 unique countries after data cleaning). As a result, the encoder's final input dimensions are (*batch_size*, 30, 956).

Similarly, the decoder takes Y as input. So we have to generate Y from Y' by concatenating the static numerical data represented by the yearly autocorrelation in our case. As a result, Y' 's dimensions which were originally (*batch_size*, 15, 779) change to (*batch_size*, 15, 780) generating Y , the actual input of the decoder.

We fine-tuned the model's parameters before training to achieve higher prediction accuracy. Figure 5 shows the learning rate fine-tuning process to select the best one which is found to be near 1e-3.

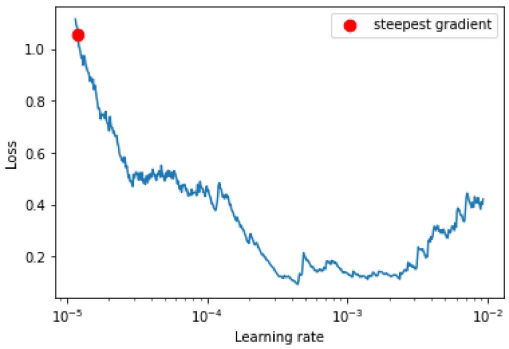


Figure 5: Learning Rate Selection from the range [1e-5, 1e-2].

A.4 Result Analysis

Table 1 shows that:

- In general, BioBERT embeddings outperform vanilla BERT embeddings for nearly all model types (with the exception of unidirectional GRU-based without attention Seq2seq and bidirectional LSTM-based without attention Seq2seq where there is a slight increase in BioBERT values) when put under the same circumstances: the truncated concatenation joining technique.
- Mean aggregation gives the top 3 results which are **0.04143** with unidirectional LSTM-based without attention, **0.04378** with bidirectional LSTM-based without attention and **0.04720** with bidirectional GRU-based without attention. Mean aggregation of BERT embeddings even outperforms the two other techniques in all occasions except for the unidirectional GRU-based without attention where the autoencoded embeddings beats the mean aggregated ones with a small factor.
- Both Mean Aggregation and Autoencoded Dimensionality Reduction are better than a simple truncated concatenation which proves our main idea that removing positional encoding from paragraph embeddings is essential when there are multiple unrelated measures at single point of time. In such cases, the order of the sentences should not be taken into account.
- It was also observed that the attention mechanism worsens the performance across all models when compared with its no-attention equivalent.
- Although the best result was produced with a unidirectional model, we found that the bidirectional option improves the results in general when compared to the unidirectional option for both with and without attention models. This can be clearly observed for all GRU-based models.
- Finally, LSTM-based models outperform GRU-based ones especially when combined with mean aggregated or autoencoded embeddings.

Table 2 shows the predictions generated by our best model on a random sample of the testing dataset. Multiple measures issued on the same date are originally concatenated with a dot. We expanded them in the table for readability purposes.

ID	Observation Date	Country	Measures	Confirmed	Predictions
896	2021-03-06	Gambia	Isolation and quarantine policies	5020	4759
324	2021-03-04	Bosnia and Herzegovina	General recommendations	141334	134899
2375	2021-02-28	Sudan	Partial lockdown	31133	30345
259	2021-02-27	Belize	Curfews Limit public gatherings Requirement to wear protective gear in public Limit public gatherings	12881	12292
58	2021-03-08	Algeria	Domestic travel restrictions	117246	113402

Table 2: Comparison between the actual and the predicted values of COVID-19 confirmed cases on a random subset of the testing data.