Deep Learning Applications in 5G RACH Procedure

Sayali Lokhande
Electrical Engg.
IIT Dharwad
Dharwad , India
200020041@iitdh.ac.in

Siba Narayan swain
Computer Science and Engg.
IIT Dharwad
Dharwad , India
sibaswain@iitdh.ac.in

I Abstract

Day-by-day, there is an increase in User Equipment(UEs) and in particular there are a lot of UEs requesting resources from eNBs(base stations). The users need to first establish connection with the eNBs to transmit information/signal. This establishment of connection involves the four step RACH procedure. The number of preambles available is 64. Since there are only a fixed number of preambles,chances of collisions among UEs are more when RACH procedure is followed. After a collision, the UE has to wait for a limited time and then restart the procedure of choosing a new preamble and establishing connection. But there is no guarantee that UE may get a unique preamble without collision. Therefore we are building some models in order to see if we can predict the time a user may request resources from eNBs and assign the preamble based on that. We are using ARIMA(Autoregressive Integrated Moving Average) and LSTM(Long-Short Term Memory) to predict the time at which a user may request resources from eNBs to avoid collisions. Here we are checking which model is working effectively and giving positive results for the datasets. Code

II Introduction

In Long Term Evolution(LTE) Networks, Random Access (RACH) is the procedure where the User Equipment (UE) wants to create an initial connection with the network. It consists of 4 signals transmitted between the UE and the base station (gNB). Firstly, UE selects a "preamble" (a code sequence) and sends it at a random time on a UL channel called Physical Random Access Channel (PRACH). UE will start monitoring the DL channel to see if the base station (eNB) answers the request to connect to the network. If not, UE will make a new attempt with the increased power. Random Access Response (RAR) sent by the network indicates which preamble it is related to, the Timing Advance (TA) it should use, a scheduling grant for sending Message 3 and a temporary Cell Radio Network Temporary Identifier(TC-RNTI).

When two or more UEs communicate the same RACH preambles to eNB, a collision occurs. Both UEs will receive the identical resource allocation at RAR (Random Access Response) step in this situation. As a result, at the L2/L3 message transmission

stage, both UE would send L2/L3 messages to the network using the same resource allocation (i.e., with the same time/frequency). There are two possibilities when each UE communicate the exact same information at the exact same time/frequency.
1) One possibility is that these two signals interfere with one another, and network is unable to decipher either of them. In this instance, none of the UEs receives a response from eNB, and they all assume the RACH procedure has failed and return to the Random Access Preamble generation step.
2) The UE with successful L2/L3 decoding on the NW side will receive the HARQ ACK from the network in this situation. The "contention resolution" process for step L2/L3 messages is referred to as the HARQ ACK process.
Thus, Message 3 and Message 4 will be used to resolve an eventual collision between 2 or more UEs attempting to access the network with the same preamble in the same physical PRACH resource, which is done by a unique identity. Once the random access (RA) procedure is completed, UE moves to connected state and UE-NW communication can continue using normal dedicated transmission.

III Prerequisites

In this section we will explain the required prerequisites to understand RACH, GAN, ARIMA and LSTM neural network

A LTE RACH

RACH stands for Random Access Channel. The purpose of the RACH is to achieve uplink synchronization between UE and eNB and to obtain the resource for message transmission.
In each LTE cell, total number of preambles availbale are 64 and UE randomly selects one of these preambles. The random access procedure in LTE is performed in the Physical Random Access Channel (PRACH), a dedicated physical channel used by UEs to request an uplink allocation from the base station.

B RACH Procedure

In the RACH Procedure, there are four phases to transmit signals between UE and the base station. The four phases are:
1) Random Access Preamble : The UEs that intend to transmit data randomly choose one among available 64 preambles and send it to the eNB.
2) Random Access Response (RAR) : The eNB processes the received signal, consisting in the superposition of all the transmitted preambles, and detects which preambles were chosen. For each detected preamble, it then sends a RAR message to assign the uplink resources to the corresponding UE.
3) L2/L3 message : The UEs use the newly assigned data channel resources to communicate their connection request, and a unique identifier.
4) Contention Resolution Message : The eNB responds to the UEs using the identifiers they communicated in their L2/L3 message, granting the requested resources.

C GAN (Generative Adversarial Networks)

Generative Adversarial Networks (GANs) are a powerful class of neural networks that are used for unsupervised learning. GANs are basically made up of a system of two competing neural network models which compete with each other and are able to analyze, capture and copy the variations within a dataset.
In GANs, there is a generator and a discriminator. The Generator generates fake samples of data(be it an image, audio, etc.) and tries to fool the Discriminator. The Discriminator, on the other hand, tries to distinguish between the real and fake samples. The Generator and the Discriminator are both Neural Networks and they both run in competition with each other in the training phase. The steps are repeated several times and in this, the Generator and Discriminator get better and better in their respective jobs after each repetition. The working can be visualized by the diagram given below:

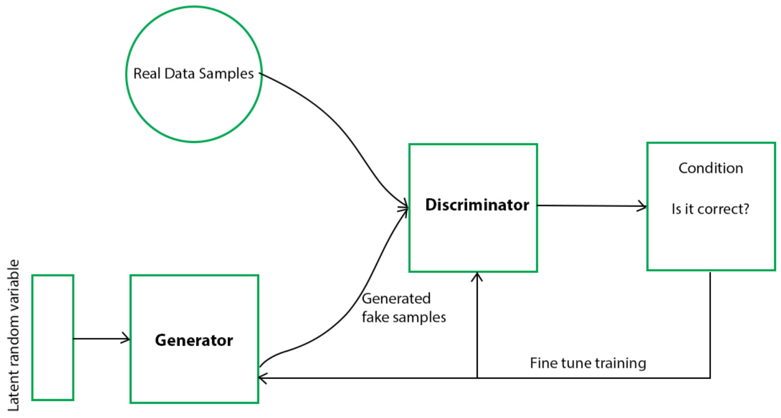

Figure 1: GAN Model

D ARIMA (Auto Regressive Integrated Moving Average)

One of the most famous and commonly used statistical models for time-series forecasting is the Auto Regressive Integrated Moving Average (ARIMA) model. It is a category of statistical techniques that captures the normal temporal dependencies that are specific to time series data.
ARIMA's different components are Auto Regressive (AR), Integrated (I), and Moving Average (MA)
AUTO REGRESSIVE(AR):
Auto Regressive (AR) model is built on top of the autocorrelation concept, where the dependent variable depends on the past values of itself.
$Y_t = \beta_1 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p}$

An observation Y at time t, Y_t, depends on Y_{t-1}, Y_{t-2}, ..., Y_{t-p} and The p here is called the lag order which indicates the number of prior lag observations we include in the model
INTEGRATED(I):
The integrated part of ARIMA attempts to convert the non-stationarity nature of the time-series data to something a little bit more stationary.
$Z_t = Y_{t+1} - Y_t$ ——— $d = 1$
$Q_t = Z_{t+1} - Z_t$ ——— $d = 2$
here $Y- > Z$ and $Z- > Q$
MOVING AVERAGE(MA):
It attempts to reduce the noise in our time series data by performing some sort of aggregation operation to your past observations in terms of residual error ϵ.
$Y_t = \beta_2 + \omega_1\epsilon_{t-1} + \omega_2\epsilon_{t-2} + ... + \omega_q\epsilon_{t-q} + \epsilon_t$
The ϵ terms represent the residual errors from the aggregation function and q here is another hyperparameter that is identical to p. But instead of identifying the time window (p) to the time series data itself, q specifies the time window for the moving average's residual error.

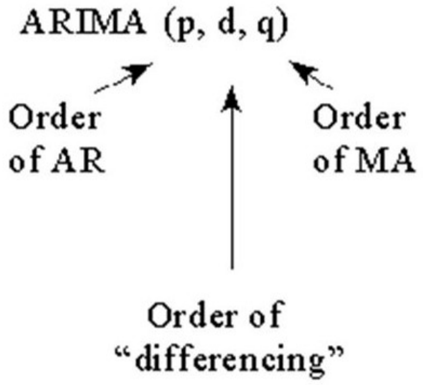

Figure 2: Basic ARIMA Model

E LSTM (Long Short Term Memory)

Long short-term memory (LSTM) is a deep learning architecture based on artificial recurrent neural networks (RNNs). LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs.

LSTM has feedback connections, unlike normal feedforward neural networks. It can handle not only single data points, but also large data sequences as well. A cell, an input gate, an output gate, and a forget gate make up a typical LSTM unit. The three gates control the flow of information into and out of the cell, and the cell remembers

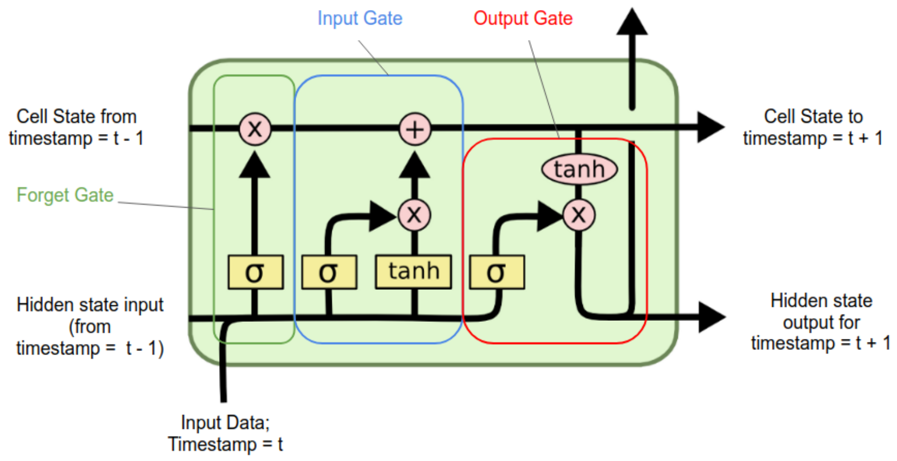
Input Gate
Output Gate
Cell State from timestamp = t - 1
Cell State to timestamp = t + 1
Forget Gate
σ
σ
tanh
σ
tanh
Hidden state input (from timestamp = t - 1)
Hidden state output for timestamp = t + 1
Input Data; Timestamp = t

Figure 3: LSTM Cell

values across arbitrary time intervals.
LSTM also has a hidden state where H(t-1) represents the hidden state of the previous timestamp and H(t) is the hidden state of the current timestamp. In addition to that, LSTM also have a cell state represented by C(t-1) and C(t) for previous and current timestamp respectively. Here the hidden state is known as Short term memory and the cell state is known as Long term memory.

IV Resource predictors

A GAN

To find the reduced number of collisions, we can group the users based on the accessed timestamps. If the same user falls in two groups, it will be considered as a collision. To calculate collisions, less data would mean less groups or less dataframes which would prove difficult in showing the model implementation especially in analysing the models based on graphs. Thus, we need every user to have atleast 1000 rows of data pertaining to timestamps. For generating synthetic data, we will use GAN model. GAN consists of a generator and a discriminator. The generator part of a GAN learns to create fake data by incorporating feedback from the discriminator. It learns to make the discriminator classify its output as real. We are defining the generator and the discriminator, generating points in latent space as input to discriminator, evaluating the discriminator and finally training both the neural networks.

B ARIMA

For predicting resource requests, we used ARIMA as an initial prediction model. Here in ARIMA model we have to find the values p,d and q where: p is the number of autoregressive terms, d is the number of non-seasonal differences needed for stationarity, and. q is the number of lagged forecast errors in the prediction equation. We are using auto-ARIMA to find the values of p, d and q. The auto-ARIMA process seeks to identify the most optimal parameters for an ARIMA model, settling on a single fitted ARIMA model. Based on the optimal values generated by auto-ARIMA for each user, the acccess time can be predicted.

C LSTM

The second predicting resource model we used is LSTM. Here we have taken one LSTM layer along with LeakyReU. LeakyReLU acts as an activation function that transforms its inputs into outputs and is defined as $f(x) = \max(0.01*x, x)$.This function returns x if it receives any positive input, but for any negative value of x, it returns a really small value which is 0.01 times x. We take first 3 values as inputs and predict the 4th value, then take next 3 values to predict the 5th value and so on. The dense layer gives the output of LSTM. Here we need only one output. We used Adam optimizer to have faster computation time. The batch size that we have used is 32 while epochs is 10 for dataset 1 while 30 for dataset 2.

V Performance Evaluation

A Datasets

The first dataset we took has all the 300 users and corresponding normalised time in a single csv file. After applying ARIMA and LSTM models and saving the predicted data along with true data in separate csv files, the amount of data is very less, thus we synthetically generated more users from given users. In the synthetic user generation, we assumed that duration and time normalized for each user comes from a bi-variate Gaussian distribution. From these, new mean and variance for new users are generated. Using this method of user generation, we generated new users by selecting different pair of users each time. Then we applied our models on these synthetically generated data.
For the second dataset we have been given with csv files for 200 users and the amount of data provided was large but the data was not normalised. Thus, we had to do some preprocessing of data to apply our models to get the predicted time.

B Prepossessing

Prepossessing refers to the addition of missing values, the creation of new features from existing features, the removal of redundant features, the standardisation or normalisation of data, and so on.
The timestamp provided with the users column in the first dataset was normalised while in the second dataset, we normalized the timestamp column by subtracting it by minimum and dividing by range (maximum - minimum).

Parameters	Dataset 1	Dataset 2
length of dataset	1000	> 7000
Training length	750	> 5250
Testing length	250	> 1750
ARIMA RMS error	0.04664772603662142	0.027524740440407
LSTM RMS error	0.005141873482298334	0.008130106163267

Table 1: Datasets

C Training

The datasets are sorted by time normalized column and then 75 % of data is used for training and remaining 25 % is used for testing for both ARIMA and LSTM model for both the datasets.

D Results

To compare the performance of LSTM based resource prediction and ARIMA based resource prediction, we took two datasets. We predicted the time at which the user selects a preamble for message transmission.

The predicted time and normalised time for 6 users of first dataset by both the models is shown below:

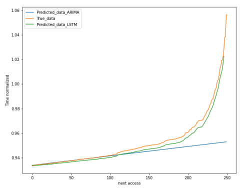

Figure 4: User 1 of dataset 1

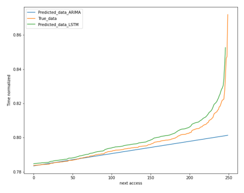

Figure 5: User 2 of dataset 1

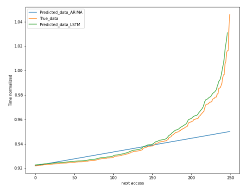

Figure 6: User 3 of dataset 1

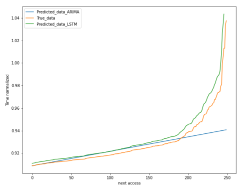

Figure 7: User 4 of dataset1

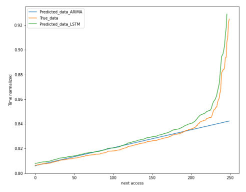

Figure 8: User 5 of dataset 1

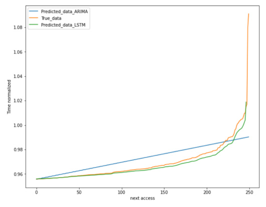

Figure 9: User 6 of dataset 1

The predicted time and normalised time for 6 users of second dataset by both the models is shown below:

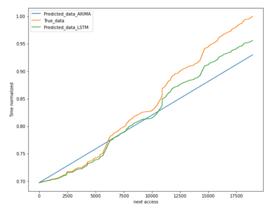

Figure 10: User 1 of dataset2

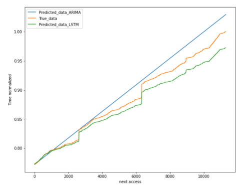

Figure 11: User 2 of dataset2

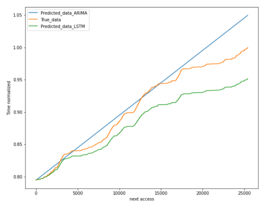

Figure 12: User 3 of dataset2

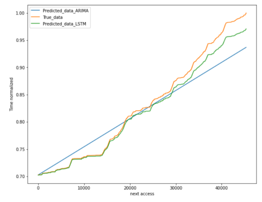

Figure 13: User 4 of dataset2

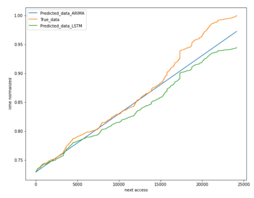

Figure 14: User 5 of dataset2

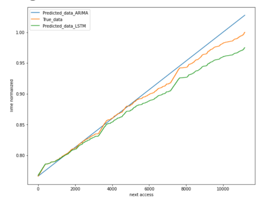

Figure 15: User 6 of dataset2

The root mean squared error obtained for both the models for the given datasets:

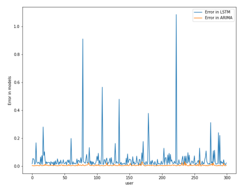

Figure 16: Error in dataset 1

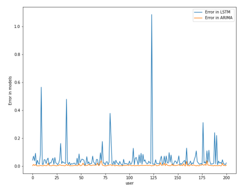

Figure 17: Error in dataset 2

VI Conclusions

We have predicted the time at which the user selects a preamble for message transmission. In this way, we can reduce the number of collisions among the users. We have shown the graphs for predicted as well as true data by 2 models, LSTM and ARIMA and also calculated the root mean squared error for them. We have shown that the mean error is less in LSTM compared to ARIMA. Thus LSTM model gives effective predictions of the time compared to ARIMA model.

References

1)S. Sesia, I. Toufik, and M. Baker, LTE, The UMTS Long Term Evolution: From Theory to Practice. Wiley Publishing, 2009.
2)G. C. Madueno, Stefanovi ˜ c, and P. Popovski, “Reengineering gsm/gprs towards a dedicated network for massive smart metering,” in ´ 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm), pp. 338–343, 2014
3)M. Cheng, G. Lin, H. Wei, and A. C. Hsu, “Overload control for machine-type-communications in lte-advanced system,” IEEE Communications Magazine, vol. 50, no. 6, pp. 38–45, 2012.
4) A. Laya, L. Alonso, and J. Alonso-Zarate, “Is the random access channel of lte and lte-a suitable for m2m communications? a survey of alternatives,”
IEEE Communications Surveys Tutorials, vol. 16, no. 1, pp. 4–16, 2014. 5) “contention resolution process.” www.sharetechnote.com/html/RACH LTE.html.
6) M. S. Ali, E. Hossain, and D. I. Kim, “Lte/lte-a random access for massive machine-type communications in smart cities,” IEEE Communications Magazine, vol. 55, no. 1, pp. 76–83, 2017.
7) O. Arouk, A. Ksentini, and T. Taleb, “Group paging-based energy saving for massive mtc accesses in lte and beyond networks,” IEEE Journal on Selected Areas in Communications, vol. 34, no. 5, pp. 1086–1102, 2016.
8) D. Magrin, C. Pielli, C. Stefanovic, and M. Zorzi, “Enabling LTE RACH collision multiplicity detection via machine learning,” CoRR, vol. abs/1805.11482, 2018.
9) “LSTM.” https://en.wikipedia.org/wiki/Long_short-term_memory
10)”ARIMA” https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average
11) ”Datasets.” https://crawdad.org/