INDIAN INSTITUTE OF TECHNOLOGY (BHU)

Department of Computer Science and Engineering
Varanasi 221005, India
July 2019

SUMMER INTERNSHIP REPORT

for

Indian Institute of Information Technology IIT-BHU
(Computer Science and Engineering Department)

by

Ambuj Krishna Agrawal

Predictive Business Process Monitoring Using LSTM and Self Attention Based Models

From **May - July, 2019** under the supervision of

Dr. K. K. SHUKLA — Professor, Department of Computer Science and Engineering IIT BHU, Varanasi.

Declaration

I certify that

1. The work contained in this report is original and has been done by myself and in the general supervision of my supervisor.

2. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to them by citing them in the text of the report.

3. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them.

Ambuj Krishna Agrawal
Place: IIT (BHU) Varanasi Btech. (2^{nd} year)
Date: July 09, 2019 Indian Institute of Information Technology, Allahabad
Information Technology (4 years)

Certificate

This is to certify that the work contained in this report entitled ***'Predictive Business Process Monitoring Using LSTM and Self Attention Based Models'*** *being submitted by* ***Ambuj Krishna Agrawal****, carried out in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, is a bonafide work of our supervision.*

Dr. K. K. Shukla
Professor
Department of Computer Science and Engineering
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Place: IIT (BHU) Varanasi
Date: July 09, 2019

Contents

Abstract

This report contains a multi-facet approach and holistic study to the predictive business process monitoring analysis. In this study, prediction for next activity, next activity's time, suffix and remaining cycle time prediction is performed. The goal was to get a better understanding of process management space with this thorough exploration. After a comprehensive comparative study of various models to accomplish the task, Long Short-Term Memory (LSTM) based neural network models with a certain number of self attention layers were found to learn the subtle intricacies of event log data with promising accuracy. Attention layers were able to capture the best time prediction in most cases while next activity prediction was done more accurately by LSTM networks with one shared layer.

Chapter 1

Chapter One

1.1 *Introduction*

What is 'predictive business monitoring' ?

Be it a gazillion dollar company like the mighty Google or any small manufacturing firm owned by a group of 'nobodies', the business requires a certain firm hand on the steering wheel with exhaustive acknowledgement of all the things that can go wrong, which includes failure of one or more intermediate stages' equipments, failure to meet deadline, failure in timely reinforcement of resources prone to exhaustion and many more such debacles. Sometimes, it becomes formidable even tending to impossible task for even a experienced human being to see through all the real world parameters and duck an impending doom, and this is where the machine learning steps in, with its mammoth number crunching abilities and memory, it has quickly become the go-to state of the art technology for this purpose. Today, predictive business monitoring models find its application in almost every maintenance requiring structure.

This whole analysis is more officially called 'Process Mining'[1]. It is basically a family of techniques in the field of process management that support the analysis of business processes based on event logs. During process mining, specialized data mining algorithms are applied to event log data in order to identify trends, patterns and details contained in event logs recorded by

an information system. Process mining aims to improve process efficiency and understanding of processes. Process mining offers objective, fact-based insights, derived from actual event logs, that help you audit, analyze, and improve your existing business processes by answering both compliance-related and performance-related questions. Process mining is categorized into following parts -

Discovery - Based on an event log, a new model is constructed or discovered based on low-level events. Many established techniques exist for automatically constructing process models based on an event log.

Conformance checking - The existing model is compared with the process event log; discrepancies between the log and the model are analyzed. Conformance checking may be used to detect deviations to enrich the model.

Enchancement - The extension or improvement of an existing process model using information about the actual process recorded in some event log.

Predictive Analysis - The model is enriched and extended with new performance information such as processing times, cycle times, waiting times, costs, etc., so that the goal is not to check conformance, but rather to improve the performance of the existing model with respect to certain process performance measures.

1.2 *Methodology*

Past information containing concerned set of events along with their corresponding usable attributes in the format is needed which can be preprocessed and fed into the model. These information is often collected with the help of hardware, software or even human units closely observing all the comprehensible details of the set of events and arrange them into usable blocks. Some important terminologies are as follows-

Process- A *process* is a series of actions or steps repeated in a progression from a defined or recognized start to a defined or recognized finish. The purpose of a process is to establish and maintain a commonly understood flow to allow a task to be completed as efficiently and

consistently as possible.

Trace - In this study we denote *trace* as the sequence of events occurred in one such instance of the complete run through of the whole process. A *trace* is a finite non-empty sequence of events such that each event appears only once and time is non-decreasing, i.e., for $1 \leqslant i < j \leqslant |\sigma|$: $\sigma(i) \neq \sigma(j)$ and $\pi_t(\sigma(i)) \leqslant \pi_t(\sigma(j))$. C is called the set of all possible traces [2].

Event log - An event log is a set of traces $L \subseteq C$ such that each event appears at most once in the entire log. It is a set of events, each linked to one trace and globally unique i.e., the same event cannot occur twice in a log. A trace in a log represents the execution of one case.

In this study only the sequence of events and the corresponding timestamp of the events for our next activity, timestamp, rest of the suffix and total time to completion of the trace prediction is taken into account. Other attributes should be acknowledged if it contains vital information that might help in prediction of future events or timestamp or other concerned parameters.

1.2.1 Next Activity Prediction

As already established that the event log dataset contains tons of traces to train and test our model. If one such trace is taken and the activity IDs corresponding to these events are $\langle 8, 4, 6, 5, 7 \rangle$, for the next activity prediction, it is said that given a prefix of specific length our model given the input would predict the next element of the sequence. For eg. if prefix length is 3, it means given the sequence $\langle 8, 4, 6 \rangle$, we predict 5 as the next activity.

1.2.2 Next Activity Time Prediction

Just like in next activity prediction, in the event log data, every activity is associated with a timestamp. Prediction has to be done for the time of the occurrence of the next event in the trace. In the previous example, prediction will be done for the occurrence of the time when

activity 5 will occur. This is done by predicting the elapsed time or the time elapsed between the future event to be predicted and the last known event timestamp. This value is simply added to the value of the last known timestamp to arrive at the new timestamp.

1.2.3 Suffix Prediction

For the suffix prediction, prefix of given length is provided and the next activity and timestamp of the event has to be predicted. Concatenating this new found data with the previous events of the trace, prediction is done for next event using the model trained for that particular prefix. It can be safely assumed that the number of future events sequence won't be more than three. This is a reasonable assumption since most of the traces are on the shorter side tending to be 2-3 in length. Prediction of the future events and timestamps are done until end of trace, which was specially introduced earlier during the preprocessing section. Below function shows similar recursive equations[2] which govern the calculation of suffix to a given activity and its corresponding timestamp.

$$f_a^{\perp}(e) = \begin{cases} \sigma & \text{,if } f_a^1(\sigma) = \perp, \\ f_a^{\perp}(\sigma \cdot e), \text{with e } \epsilon \text{ E }, \pi_A(e) = f_a^1(\sigma) & \\ \wedge \pi_T(e) = (f_t^1(\sigma) + \pi_T(\sigma(\mid \sigma \mid))) & \text{,otherwise} \end{cases} \tag{1.1}$$

$$f_t^{\perp}(e) = \begin{cases} \sigma & \text{,if } f_t^1(\sigma) = \perp, \\ f_t^{\perp}(\sigma \cdot e), \text{with e } \epsilon \text{ E }, \pi_A(e) = f_a^1(\sigma) & \\ \wedge \pi_T(e) = (f_t^1(\sigma) + \pi_T(\sigma(\mid \sigma \mid))) & \text{,otherwise} \end{cases} \tag{1.2}$$

In order to evaluate the results with a metric we use 'Damerau-Levenshtein Similarity' (DLS). Damerau-Levenshtein distance is defined as the minimum number of operations needed to convert one string into another (predicted suffix and ground truth suffix) where the operations can be of four types namely insert, replace, swap and delete. This number is calculated using dynamic programming which takes O(m*n) time and takes space O(m*n) to do it where m and

n are the length of the concerned pair of strings. This number is divided with the maximum of the length of predicted suffix and ground truth suffix. When subtracted with 1, gives the DLS value[2].

1.2.4 Remaining Cycle Time Prediction

The remaining cycle time prediction is an extension to the suffix prediction where the process of predicting future events is repeated until the end of case is encountered. The corresponding timestamp is called the end time of that trace. Hence the remaining cycle time would be the difference between the end time and the current time or the last recorded time[2].

1.2.5 Baseline Methods

The baseline methods as demonstrated in the paper[3] have the property of statistically predicting the next event and timestamp data. Hence for every model claiming to predict these attributes should have error less than the baseline methods.

Baseline methods serve to look at our data with a different perspective or with a different abstraction than usual which helps to categorize them on the basis of the abstraction used. We make a transition system, which given a event transforms the given or previously known abstracted attributes to new future events which have to be predicted.

1) Set abstraction - In this type of abstraction, only the unique sorted events of the trace are considered to predict the next event.

2) Sequence abstraction - In this abstraction, the whole sequence is considered as it is, which is arguably the least of all the abstractions but it does help in prediction of future attributes.

3) Bag abstraction - In bag abstraction, multi set representation is used in which the event along with it's number of occurrence is stored in multiset format. Other than that only the unique elements are taken but the order is not disturbed along with its cardinality.

For eg. let a trace be $\langle 4, 8, 4, 9, 1, 4 \rangle$, the set abstracted version of this trace would be $\{4,8,9,1\}$.

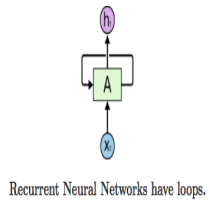


Figure 1

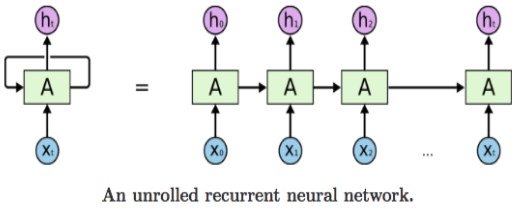


Figure 2

Similarly for sequence abstraction, the trace would look like $\langle 4, 8, 4, 9, 1, 4 \rangle$. The bag abstracted form would look like a multiset $[4^3, 8, 9, 1]$.

1.3 *Recurrent Neural Networks*

Since humans don't start their thinking from scratch every second, derivation has to be done from previously received knowledge and it has to be correlated to that information to understand current and upcoming words or sequence.
Simple neural network lack this ability to understand from previous information or sequence where as RNN inputs information from input sequence as well as previously learned information from that sequence, hence it finds its application in myriads of field like speech recognition, language modeling, translation, image captioning and many more.

RNN model inputs sequence of a fixed size but the output can be of variable length which is in accordance with the need of the model in the first place. Let's say, there is a input to the model, a sentence, each word at a time with each word having its own vector to denote the characteristics of that word, this outputs a particular sequence of numbers denoting the probability that the sentence falls into a given domain.

Figure 1 below shows an looped version of RNN where the predicted values are again input along with the optional input from original sequence(X), to predict next value of the sequence or any of it's attribute. Where as Figure 2 is the unrolled version which shows the same concept with much better depiction of the above process[4].

1.4 *Long Short-Term Memory*

RNN has the ability to understand from previous sequence to predict the future but it lacks from what is called the 'vanishing gradient' problem. During the backpropagation multiplication is done on the current parameters with a certain adjustments to predict the goal better. When the number of elements in the sequence is large these adjustments keep getting smaller as earlier sequence are considered and hence, the information in the form of weights value decreases and becomes negligible. Hence, the ability of the model to understand from sequences far behind the current input decreases exponentially. This is where LSTM or long short-term memory models come in which specializes in capturing long term dependencies. For eg.if there is a sentence, 'The birds were so happy and chirping that it felt like they were going to lay eggs'. It would be very useful to know that the word 'were' is actually used because birds are plural.

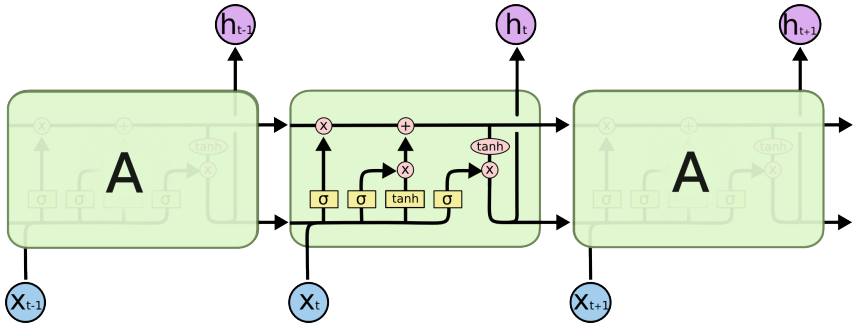


Figure 3. *LSTM cells [4]*

The core idea behind LSTM is the existence of memory or state cells beside the usual RNN structure which stores the usable information in the memory implemented using various gates namely, forget gate, input gate and lastly the output gate.

How this works is that a suitable candidate is found using the formulas below. Preparation of I(t) and F(t) is done to calculate the replacement and find the new output using this replacement and churn out candidate value and new output also using O(t) i.e output gate. Backpropagation is used to train these weights often called 'backpropagation through time' .

Forget Gate:- A forget gate is responsible for removing information from the memory unit or cell state. The information that is no longer required for the LSTM to understand things or the information that is of less importance is removed via multiplication of a filter.

Input Gate:- The input gate is responsible for the addition of information to the cell state. This information is also passed onto the subsequent cells as information. It is supposed to incorporate only the necessary information.

Output Gate:- The job of selecting useful information from the current cell state and showing it out as an output is done via the output gate[5].

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{1.3}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{1.4}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{1.5}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{1.6}$$

$$h_t = o_t \cdot \sigma_h(c_t) \tag{1.7}$$

1.5 *Attention*

The most fascinating thing about models in machine learning is that it is often inspired from real life setting capturing the most subtle of intuitions. Just like a human processes an image, it automatically focuses on specific parts of an image and not the whole part, courtesy to the years of training on focusing on the important parts of human vision. The attention layer does a similar job by learning to focus only on the important parts of the previously encountered sequence hence increasing accuracy in sequence data. It works wonders in machine language translation as written in the paper, ”Attention Is All You Need”[6].

For eg. in image captioning if the model is aware where to give more emphasis in deciding the activity then automatically the accuracy will increase since the noise introduced due to irrelevant information is reduced.

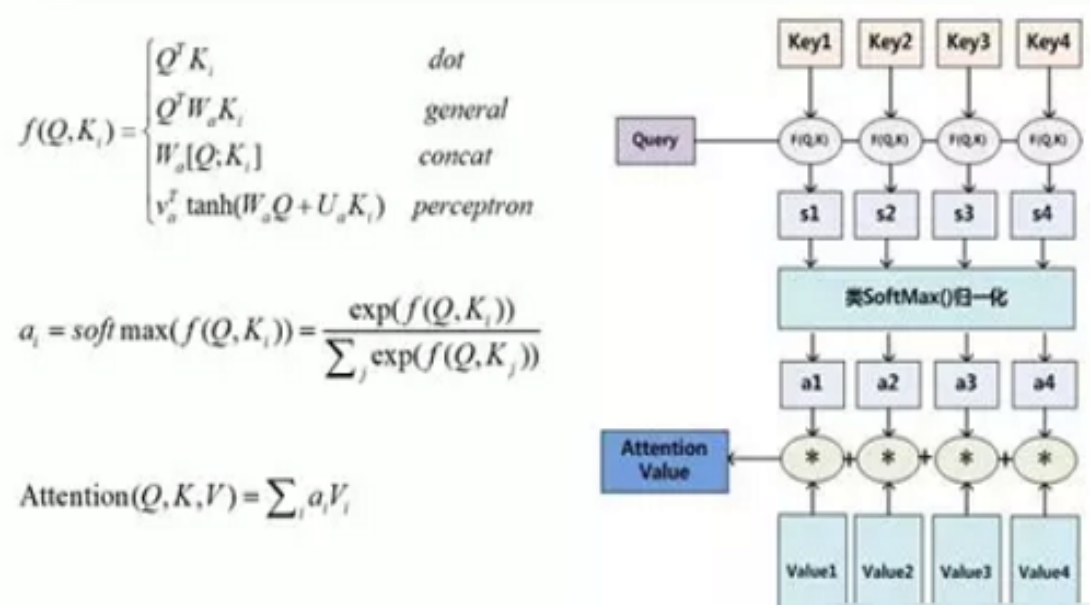


Figure 4. *Attention Working*

How this works is that, the embeddings of the input sequence is transformed into query, key and value pairs whose length is in accordance with the length of the input sequence. Then for each query, the corresponding query vector is operated with all the key vectors according to the types of operations as mentioned in Figure 4[7], dot product being the most common one. The produced values are run through the softmax unit and the values produced are called 'attention values'. These values are multiplied with the corresponding value pair to produce the input sequence length number of vectors. These vectors are simply concatenated to give final values for that query vector through the attention layer. This process is repeated for all the query vectors. Each output not only contains the information about that input but the usable information from the past relevant sequence as well. In this report, a variation called multi-headed self attention is used which gives better results since multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions.

Chapter 2

Chapter Two

2.1 *Dataset*

2.1.1 Helpdesk

This log contains events from a ticketing management process of the help desk of an Italian software company. The process consists of 9 activities, and all cases start with the insertion of a new ticket into the ticketing management system. Each case ends when the issue is resolved and the ticket is closed. This log contains around 3,804 cases and 13,710 events.

2.1.2 BPI'12 (no duplicate)

This event log originates from the Business Process Intelligence Challenge (BPI'12) and contains data from the application procedure for financial products at a large financial institution. This process consists of three subprocesses: one that tracks the state of the application, one that tracks the states of work items associated with the application, and a third one that tracks the state of the order. We narrow down our evaluation to the work items subprocess, which contains events that are manually executed.

For our study, a variation of the above dataset was used in which all the duplicate events in a trace were removed so as facilitate the suffix and remaining time prediction. For our report, the dataset had 9658 events with 6 different types of activities.

Chapter 3

Chapter Three

3.1 *Preprocessing*

3.1.1 Feature Extraction

The data received was in .xes format which was converted into .csv format with 1000s of traces each containing variable number of events. Since preprocessing of the data such that it could be an input to the LSTM or attention layers. The first objective was extract features from the data which might be vital in deciding the next event. The columns of the .csv file was reduced (if not already) to contain only the trace number (CaseID), ActivityID and the timestamp of the event.

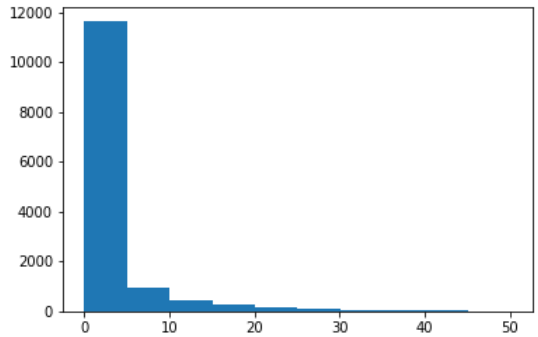


Figure 5. *elapsed time(1) hist plot*

The features used in this analysis are inspired from the research paper[2]. The first feature was found out to be the elapsed time or the time between the previous event in the trace and the current one. If the event in consideration is actually the first event of the trace, then the elapsed time is considered to be zero. π_{T}(e) implies the projection of the event e of the trace considering only the Timestamp value. While $\pi_{\mathrm{T}}(\sigma(i-1))$ implies the time attribute of the event number i-1 of the trace σ. The following equation[2] summarizes it.

$$fv_{t1}(e) = \begin{cases} 0 & \text{if } i = 1, \\ \pi_{\mathrm{T}}(e) - \pi_{\mathrm{T}}(\sigma(i-1)), & \text{otherwise} \end{cases} \tag{3.1}$$

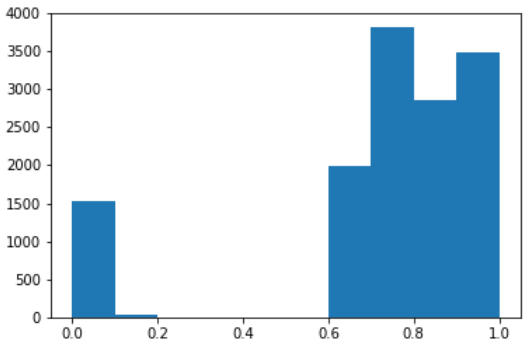


Figure 6. *From midnight time(2) hist plot*

The second feature was found to be the time of the event from midnight. It was done to help understand model the working schedule of these events on a random day.

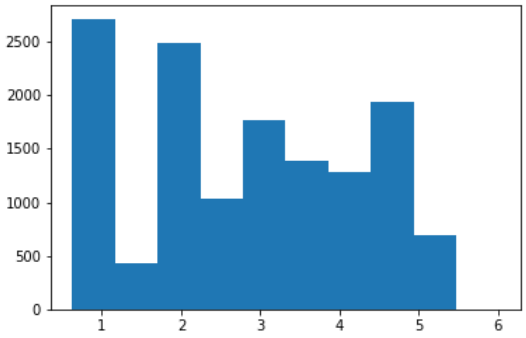


Figure 7. *From last sunday time(3) hist plot*

The third feature was found to be the duration between the current timestamp and the last sunday midnight. It was done so as to help understand the model of the general working days of the event to help predict better results.
Above features were first normalized and converted into days for easy prediction units. Also they were calculated for each event of each trace and stored as a column of the Dataframe used as an interface of the .csv file.

	CaseID	ActivityID	datetime	elap	from_midnight	from_sun
0	2	1	2012-04-03 16:55:38	0.000000	0.705301	1.705301
1	2	8	2012-04-03 16:55:53	0.000174	0.705475	1.705475
2	2	6	2012-04-05 17:15:52	2.013877	0.719352	3.719352
3	3	1	2010-10-29 18:14:06	0.000000	0.759792	4.759792
4	3	8	2010-11-04 01:16:11	5.293113	0.052905	3.052905

Figure 8. *Current data snippet*

For the helpdesk data, maximum of 9 unique events were found to be present, hence we find the one-hot encoding of the maximum length of 10 where we introduce an event (0) which denotes the end of the trace.

For the BPI'12 (no duplicate) data, maximum of 6 unique events were found to be present, hence we find the one-hot encoding of the maximum length of 7 where introduce an event (0) which denotes the end of the trace just like in helpdesk data.
Concatenating the above features makes it total of 13 events for each trace in helpdesk data and 10 for each trace in BPI'12 data.

3.1.2 Prefix Length Adjustments

The data in helpdesk was found to contain maximum of 14 events in a trace. Similarly for BPI'12 data, this number was found to be 13 events. To make things easier, we iterate through each trace in the respective dataset and post-pad them to the maximum number of events in that trace (14 and 13 respectively). The shapes after padding comes out to be (3804,14,13) for

helpdesk and (9658,13,10) for BPI'12 dataset. Above data is divided such that first 4/5th of data is for training and the last 1/5th data is for testing.

Now, let's say the prefix required is of length 3. For the training data we extract our X_train as (0.8*3804,:3,13) and Y_train as (0.8*3804,3,:10) for activity prediction and (0.8*3804,3,10) for timestamp (elapsed time) prediction.

3.1.3 Baseline Methods

A transition system was established in a similar way as inspired by the research paper [3]. with the same dataset for comparision purposes with the LSTM and attention models using dictionary in python. It used statistical measure in the prediction of next activity and time prediction. After applying the concerned abstraction, similar traces were grouped together. The most occurring next activity was declared as the next activity for that particular sequence obtained after applying abstraction. If an unknown input sequence occurs, the model simply predicts the most common next activity as the answer. For next timestamp prediction, similar traces were again grouped together and average of the property 'elapsed time', was taken and was stated as the final elapsed time for that particular sequence, if occurred in test data. For unknown future sequences the model simply predicts the average of all the elapsed time in the dataset. This process was applied for each of three abstractions namely, set, sequence and bag.

3.2 *Model Architecture*

As addressed in the paper[2], the model can be designed in several ways according to the need and relevance. Figure 9. shows such three types of designs. In the figure (a), layers are single task i.e they don't share information with some common previous layer and are independently tuned and trained where as figure (b) has shared multitasking layers i.e layers share information with each other and are jointly trained and tuned. This specially works wonders when the status of one type of prediction have some parameters which affects the other prediction as well.

Figure(c) is an advance version of figure(b) where there are multiple shared and multiple single tasking layer.

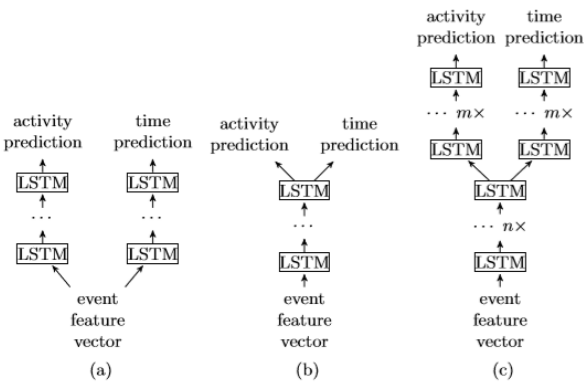


Figure 9. *Neural Network architectures with single-task layers (a), with shared multitasks layer (b), and with n + m layers of which n are shared (c)*

3.2.1 No shared layer

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 2, 100)            45600
_________________________________________________________________
dropout_1 (Dropout)          (None, 2, 100)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 100)               80400
_________________________________________________________________
batch_normalization_1 (Batch (None, 100)               400
_________________________________________________________________
dropout_2 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1010
=================================================================
Total params: 127,410
Trainable params: 127,210
Non-trainable params: 200
_________________________________________________________________
```

Figure 10. *Activity prediction with no shared layer*

Figure 10 shows the architecture of the model with no shared layer for only activity prediction where as Figure 11 shows the one with no shared layer only for elapsed time prediction. This type of design is similar to Figure 9(a). Both the models have two LSTM units stacked on top

of each other with each having 100 hidden units. Activity prediction model outputs 10 softmax results denoting the probable activity for helpdesk data where as time prediction model predicts only the elapsed value for the next activity so that it could be added to last recorded activity time and the new timestamp could be obtained.

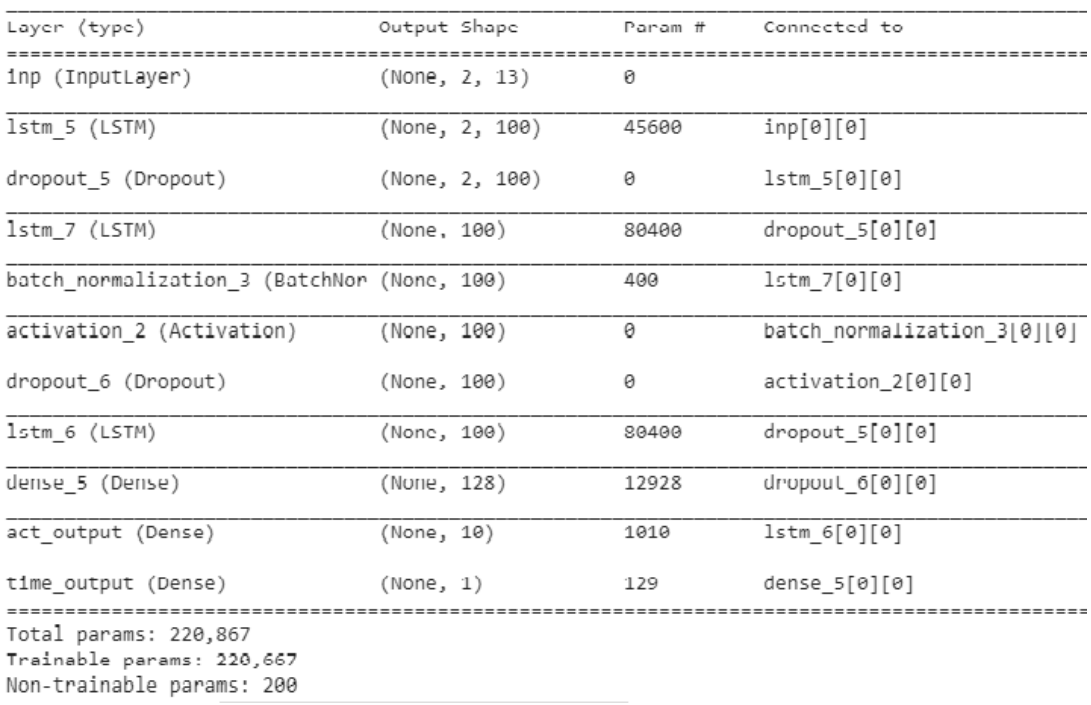
Layer (type)	Output Shape	Param #
lstm_3 (LSTM)	(None, 2, 100)	45600
dropout_3 (Dropout)	(None, 2, 100)	0
lstm_4 (LSTM)	(None, 100)	80400
batch_normalization_2 (Batch	(None, 100)	400
activation_1 (Activation)	(None, 100)	0
dropout_4 (Dropout)	(None, 100)	0
dense_2 (Dense)	(None, 128)	12928
dense_3 (Dense)	(None, 32)	4128
dense_4 (Dense)	(None, 1)	33

Total params: 143,489
Trainable params: 143,289
Non-trainable params: 200

Figure 11. *Timestamp prediction with no shared layer*

3.2.2 Shared layers

Figure 12 shows the architecture of the model with one shared layer giving two outputs, the 10 sized activity softmax and one elapsed time prediction. Two LSTM units are used each with 100 hidden units in which one of them is common to both prediction which learns from both the losses and tries to learn from it. The next layer is specific for both activity and time, intuitively it acts as a specialized layer for a specific job which here is activity and time. The different second layer in both learns the intricacies specific to that prediction.



Layer (type)	Output Shape	Param #	Connected to
inp (InputLayer)	(None, 2, 13)	0	
lstm_5 (LSTM)	(None, 2, 100)	45600	inp[0][0]
dropout_5 (Dropout)	(None, 2, 100)	0	lstm_5[0][0]
lstm_7 (LSTM)	(None, 100)	80400	dropout_5[0][0]
batch_normalization_3 (BatchNor	(None, 100)	400	lstm_7[0][0]
activation_2 (Activation)	(None, 100)	0	batch_normalization_3[0][0]
dropout_6 (Dropout)	(None, 100)	0	activation_2[0][0]
lstm_6 (LSTM)	(None, 100)	80400	dropout_5[0][0]
dense_5 (Dense)	(None, 128)	12928	dropout_6[0][0]
act_output (Dense)	(None, 10)	1010	lstm_6[0][0]
time_output (Dense)	(None, 1)	129	dense_5[0][0]

Total params: 220,867
Trainable params: 220,667
Non-trainable params: 200

Figure 12. *Activity and Timestamp prediction with one shared layer*

3.2.3 Attention layers

Attention layer is simply an addition to the previous shared layer architecture. As explained previously, attention captures only relevant data from before and helps in prediction suffices with better results. In this model multi-headed self attention layer with two heads are used and then the model is branched into its two constituents namely activity and time prediction.

Layer (type)	Output Shape	Param #	Connected to
inp (InputLayer)	(None, 2, 13)	0	
lstm_33 (LSTM)	(None, 2, 100)	45600	inp[0][0]
attention_6 (Attention)	(None, 2, 192)	57600	lstm_33[0][0] lstm_33[0][0] lstm_33[0][0]
dropout_30 (Dropout)	(None, 2, 192)	0	attention_6[0][0]
batch_normalization_15 (BatchNo	(None, 2, 192)	768	dropout_30[0][0]
activation_12 (Activation)	(None, 2, 192)	0	batch_normalization_15[0][0]
dropout_31 (Dropout)	(None, 2, 192)	0	activation_12[0][0]
attention_5 (Attention)	(None, 2, 192)	57600	lstm_33[0][0] lstm_33[0][0] lstm_33[0][0]
dense_21 (Dense)	(None, 2, 128)	24704	dropout_31[0][0]
flatten_6 (Flatten)	(None, 384)	0	attention_5[0][0]
flatten_5 (Flatten)	(None, 256)	0	dense_21[0][0]
act_output (Dense)	(None, 10)	3850	flatten_6[0][0]
time_output (Dense)	(None, 1)	257	flatten_5[0][0]

Total params: 190,379
Trainable params: 189,995

Figure 13. *Activity and Timestamp prediction with one shared layer and one attention layer*

Above architectures are for prefix length 2 of helpdesk data.

3.3 *Results*

3.3.1 Helpdesk predictions

For following graphs, helpdesk data with prefix = 2 and validation percent of 20 is taken.

3.3.1.1 Activity Prediction

Figure 14 shows the losses results plotted into a line curve for better understanding. Clearly even single tasking layer is performing quite decently since both losses are continuously decreasing and the difference between the training and validation data is minimal denoting the absence

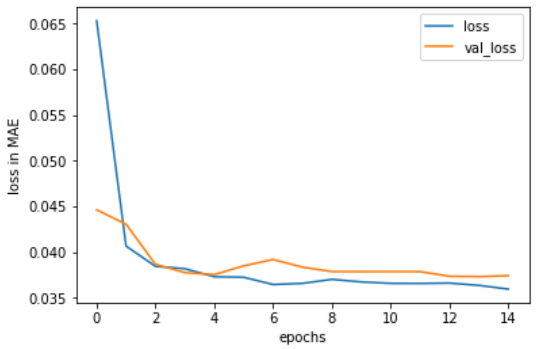


Figure 14. *Activity prediction with no shared layer losses(training and validation)*

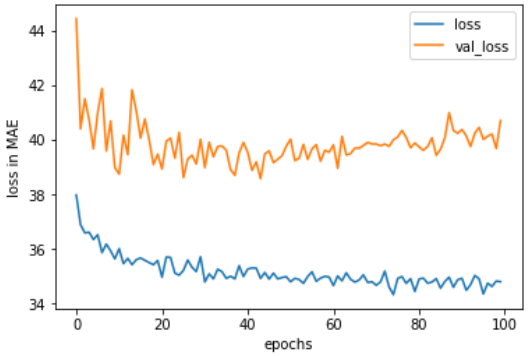


Figure 15. *Elapsed time prediction with no shared layer losses(training and validation)*

of overfitting or underfitting. The accuracy was found out to be **78.3%** on test data which constituents last 20% of the traces in the dataset.

3.3.1.2 Time Prediction

Figure 15 shows the elapsed time prediction, which clearly suffers from overfitting since the difference between the training loss and validation loss is quite a lot. This was kind of expected since given the length of prefix and low number of traces for the model to learn, it won't generalize on unknown traces nicely as of yet. The training data continues to overfit while the validation after getting an all time low starts increasing as we go for longer epochs. A technique called 'Early Stopping' is used to capture the best scenario with minimum difference between the two losses. The losses are calculated in MAE (mean absolute error) in terms of days since earlier we converted everything into days.With no shared layer the result showed loss of **3.744** MAE.

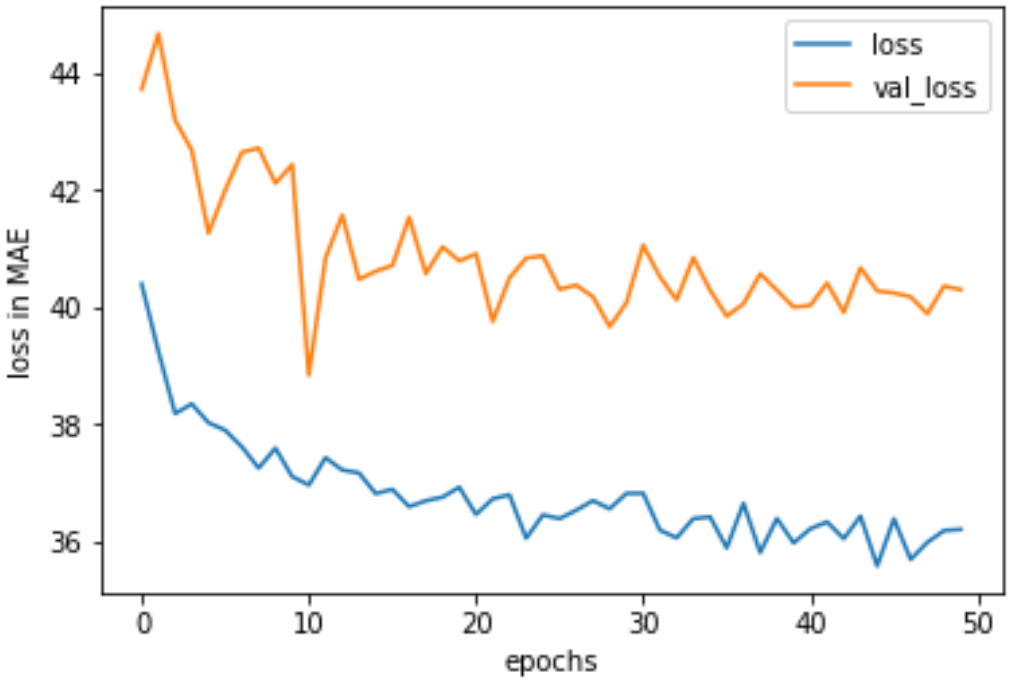


Figure 16. *Shared layer activity and elapsed time prediction losses for training and validation data*

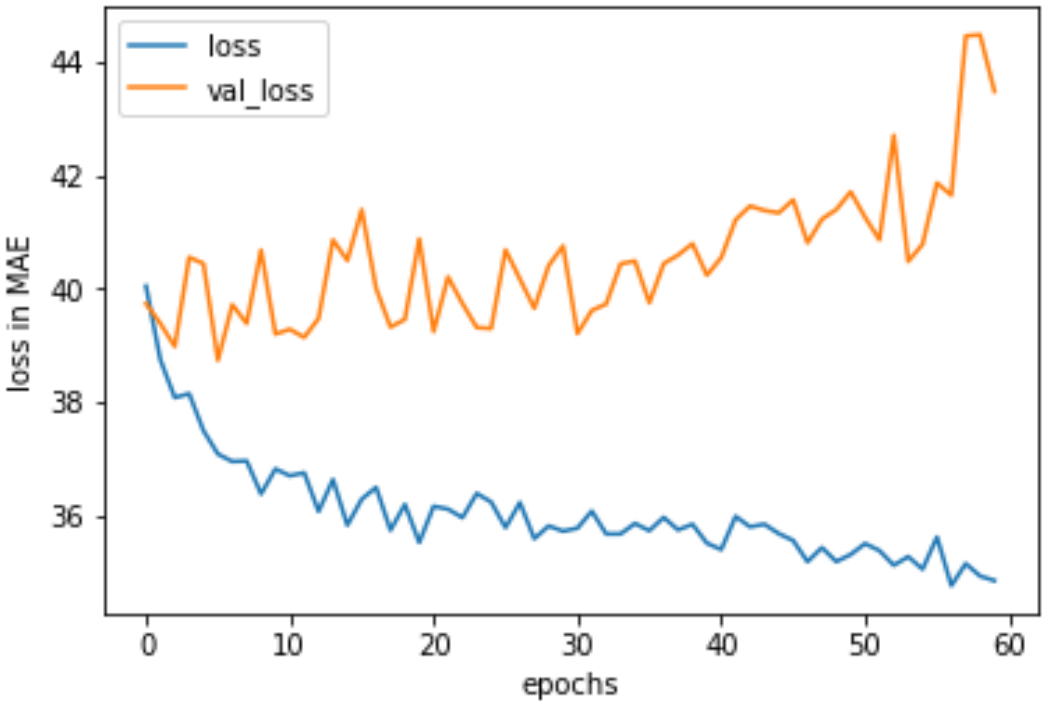


Figure 17. *Shared layer+attention layer, activity and elapsed time prediction losses for training and validation data*

Figure 16 shows the activity and time prediction for the shared layer scenario. Clearly the plot is quite similar to the 'single tasking' layer time prediction since the activity prediction

Table 1. *Helpdesk predictions*

Model	Activity Prediction(accuracy %)			Time Prediction(MAE in days)		
	2	4	6	2	4	6
0 shared layer	78.3	84.0	99.0	3.45	1.74	1.38
1 shared layer	78.4	84.0	97.5	3.401	1.61	1.21
1 shared and 1 attention	78.4	83.5	97.1	3.315	1.43	2.21
set				3.92	5.67	5.55
sequence				3.92	5.84	5.84
bag				3.92	5.84	5.46

generates really minimal loss. 'Early Stopping' is again used to capture the best scenario where the validation loss is least. Activity prediction has accuracy of **78.4** and time MAE of **3.452** in days.

In figure 17, we have attention layer model which again suffers from overfitting since the number of trainable parameters are huge and number of training examples are quite less. We can notice a small dip at around 20 epochs which we capture through 'Early Stopping'. Activity prediction has accuracy of **78.4 %** and time of **3.401** MAE in days.

All the three models clearly performed better than the baseline methods. The baseline predictions were exceptionally less for prefix 2. All the models clearly showed better results in terms of accuracy and MAE as we increased the prefix length. Test data was separated initially as 20% of the complete original data. The results on test data along with the baseline predictions are compiled in Table 1.

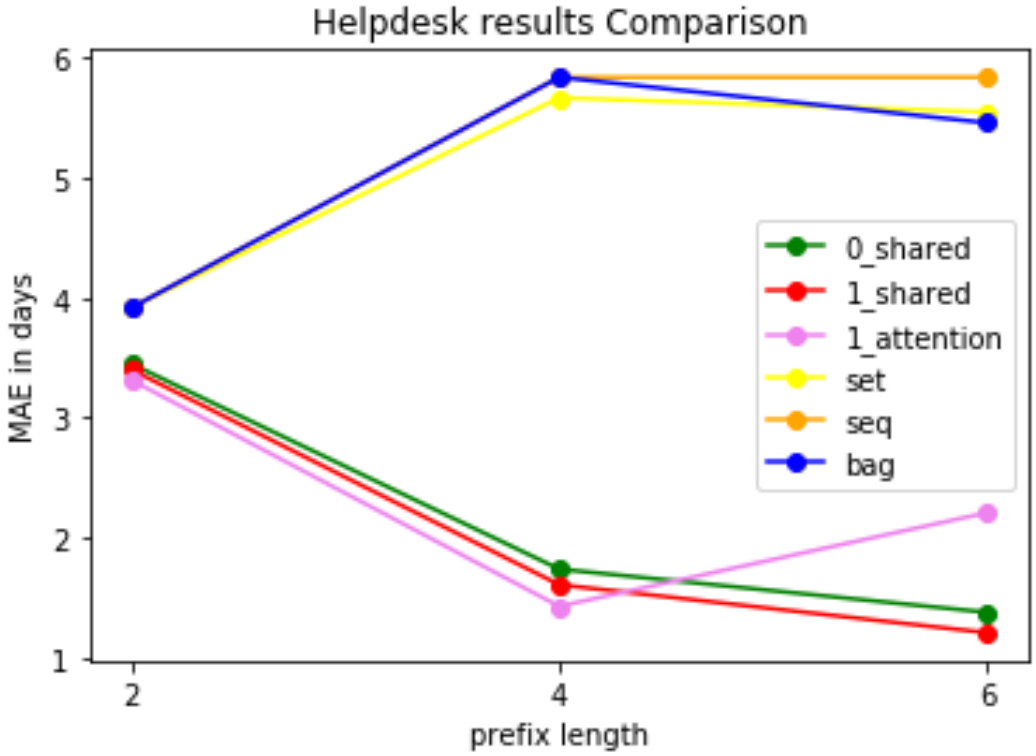


Figure 18. *Trends shown by helpdesk data*

3.3.1.3 Suffix Prediction

Calculation of normalized Damerau-Levenshtein distance is done for each trace in the test data. Then the average of all the these values are taken and subtracted from 1 to give the Damerau-Levenshtein Similarity (DLS) array. For the example, prefix length of 2 was taken and the results were promising. For the test data, it gave Damerau-Levenshtein Similarity of **81.60**.

3.3.1.4 Remaining Cycle Time Prediction

For each trace in suffix prediction, final timestamp was calculated for each event in suffix. The timestamp for which next event is 0 denoting the end of trace is separated and difference is taken from last known timestamp. Then average is taken for all such values in the test data. This process was done for prefix length 2 and result was **5.26** MAE in days while for prefix length 3 is **3.34**. Clearly for prefix 3 it is smaller which makes sense. Also the difference is

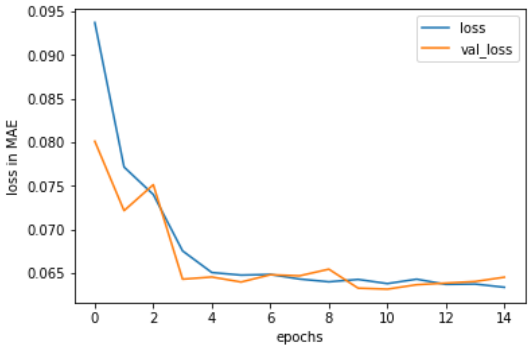


Figure 19. *Activity prediction with no shared layer losses(training and validation)*

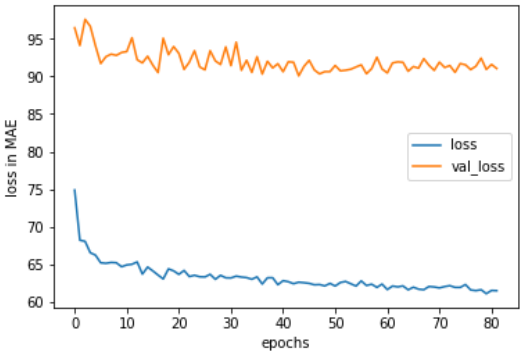


Figure 20. *Elapsed time prediction with no shared layer losses(training and validation)*

considerable because the model is designed such that it predicts suffix of maximum length 4 from prefix length 2.

3.3.2 BPI'12 (no duplicates) predictions

For below graphs, BPI'12 (no duplicates) data with prefix = 2 and validation percent of 20 is taken.

3.3.2.1 Activity Prediction

Figure 19 shows the losses results plotted into a line curve for better understanding. Clearly even single tasking layer is performing quite decently since both losses are continuously decreasing and the difference between the training and validation data is minimal denoting the absence of overfitting or underfitting. The accuracy was found out to be **67.3%** on test data which constituents last 20% of the traces in the dataset.

3.3.2.2 Time Prediction

Figure 20 shows the elapsed time prediction, which clearly suffers from overfitting since the difference between the training loss and validation loss is quite a lot. This was kind of expected since given the length of prefix and low number of traces for the model to learn, it won't generalize on unknown traces nicely as of yet. The training data continues to overfit while

the validation after getting an all time low starts increasing as we go for longer epochs. A technique called 'Early Stopping' is used to capture the best scenario with minimum difference between the two losses. The losses are calculated in MAE (mean absolute error) in terms of days since earlier we converted everything into days. With no shared layer the result showed loss of **5.43** MAE.

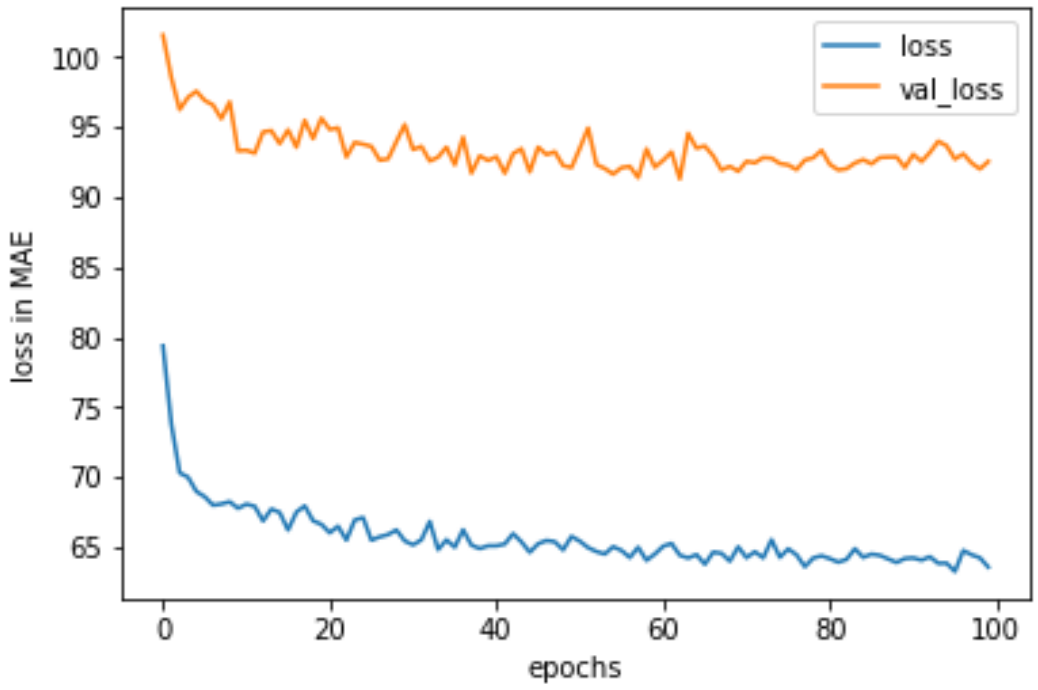


Figure 21. *Shared layer activity and elapsed prediction losses for training and validation data*

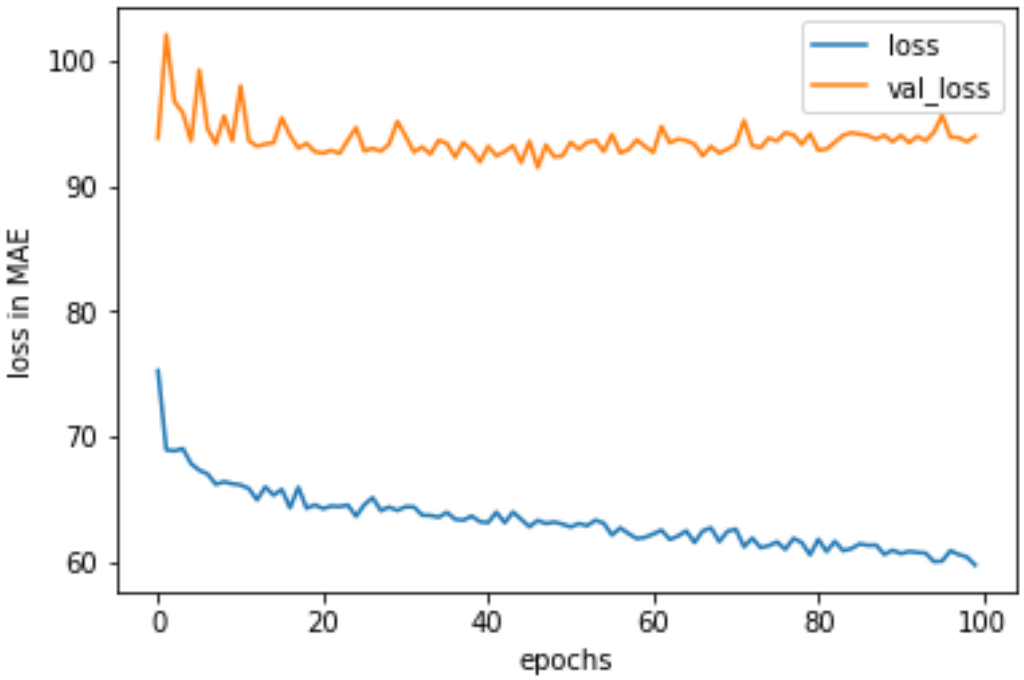


Figure 22. *Shared layer+attention layer, activity and elapsed time prediction losses for training and validation data*

Figure 21 shows the activity and time prediction for the shared layer scenario. Clearly the plot is quite similar to the 'single tasking' layer time prediction since the activity prediction generates really minimal loss. Again 'Early Stopping' is used to capture the best scenario where the validation loss is least. Activity prediction has accuracy of **67.4** and time MAE of **5.39** days. In figure 21, we have attention layer model which again suffers from overfitting since the number of trainable parameters are huge and number of training examples are quite less. We can notice a small dip at around 40 epochs which we capture through 'Early Stopping'. Activity prediction has accuracy of **70.1** and time MAE of **5.30** days.

The baseline predictions were exceptionally less for prefix 2. It was because of such a short prefix that the transition system implemented using dictionary in python could cluster similarly abstracted traces together, hence the time prediction was unfairly accurate which the model couldn't learn. For other prefixes, the general trend followed and the results for the machine learning models were better than baseline methods. All the models showed better prediction in terms of accuracy and MAE as we moved towards larger prefixes, since more history was

Table 2. *BPI'12 predictions*

Model	Activity Prediction(accuracy %)			Time Prediction(MAE in days)		
	2	4	6	2	4	6
0 shared layer	67.3	75.9	58.9	5.39	1.25	0.79
1 shared layer	67.4	76.1	59.9	5.35	1.06	0.83
1 shared and 1 attention	70.1	68.02	52.1	5.23	1.05	0.75
set				6.57	7.67	2.727
sequence				6.57	7.65	2.88
bag				6.57	7.65	2.80

available for model to learn. Test data was separated initially as 20% of the complete original data. The results on test data along with the baseline predictions are compiled in Table 1.

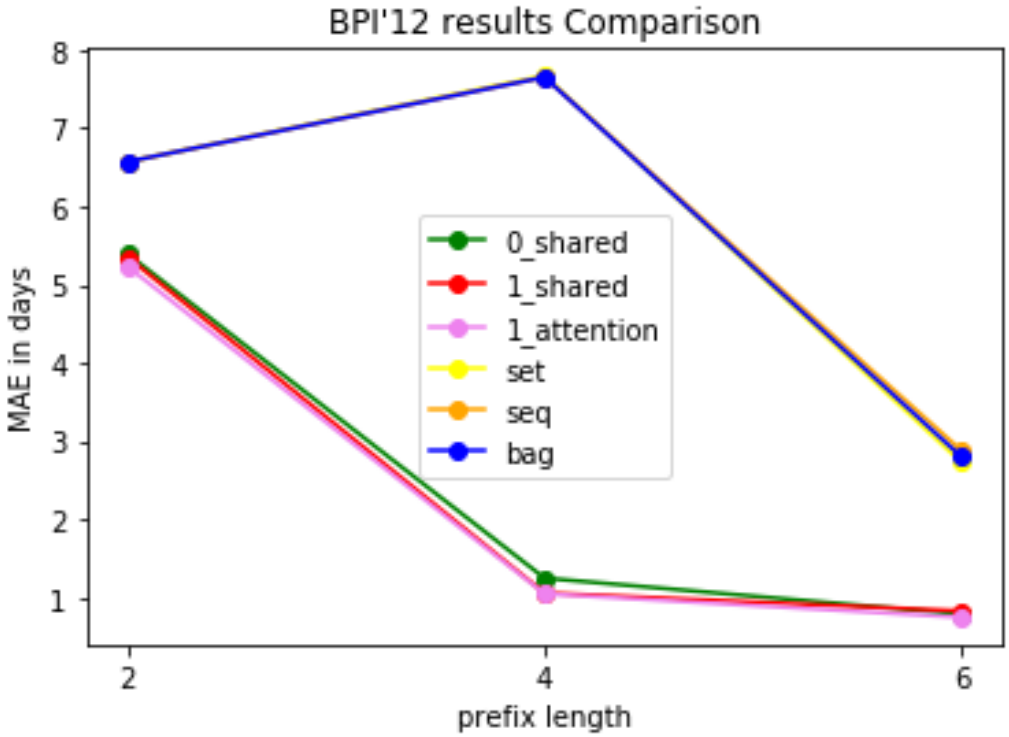


Figure 23. *Trends shown by BPI'12*

3.3.2.3 Suffix Prediction

Calculation of normalized Damerau-Levenshtein distance is done for each trace in the test data. Then the average of all the these values are taken and subtracted from 1 to give the

Damerau-Levenshtein Similarity (DLS) array. For our example, prefix length of 2 and the results were promising. For the test data, it gave Damerau-Levenshtein Similarity of **48.24**.

3.3.2.4 Remaining Cycle Time Prediction

For each trace in suffix prediction, final timestamp was calculated for each event in suffix. The timestamp for which next event is 0 denoting the end of trace is separated and difference is taken from last known timestamp. Then average is taken for all such values in the test data. This process was done for prefix length 2 and result was **8.24** MAE in days where as for prefix length 3 it gave **8.002** MAE in days.

Conclusion & Future Work

The trend of process management has been reaching new heights in recent years owning to its revolutionary usage in maintenance of any form of structure with exquisite attention to seemingly abstract details. The models, specifically shared layer Long short-term memory models and Attention layer models works decently and gave good results for the helpdesk and BPI'12 dataset as researched upon extensively in this report.

Current research explores usage of clustering algorithms along with feature hashing and auto encoders to extract complicated features and predict next activity and time in a event log[8]. It's been only till recently that Generative Adversarial Network (GAN) were used in the domain of natural language processing. It shows very promising results in time series data because of its characteristics of adapting to a probability distribution of sequence of data. If applied to a suitably large dataset, it could work wonders in this field. Though, it might be a daunting task.

Appendix

The github link for the code can be found at

https://github.com/aka-jumba/predictive_process_management

Bibliography

[1] Wikipedia. *Process mining — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Process%20mining&oldid=902673339. [Online; accessed 03-July-2019]. 2019.

[2] Niek Tax et al. "Predictive business process monitoring with LSTM neural networks". In: *International Conference on Advanced Information Systems Engineering*. Springer. 2017, pp. 477–492.

[3] Wil MP Van der Aalst, M Helen Schonenberg, and Minseok Song. "Time prediction based on process mining". In: *Information systems* 36.2 (2011), pp. 450–475.

[4] *Understanding LSTM Networks*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[5] Pranjal Srivastava. *Essentials of Deep Learning : Introduction to Long Short Term Memory*. 2017. URL: https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/.

[6] Ashish Vaswani et al. "Attention Is All You Need". In: *NIPS*. 2017.

[7] /@Alibaba$_{Cloud}$. *Self-Attention Mechanisms in Natural Language Processing*. 2018. URL: https://medium.com/@Alibaba_Cloud/self-attention-mechanisms-in-natural-language-processing-9f28315ff905.

[8] N. Mehdiyev, J. Evermann, and P. Fettke. “A Multi-stage Deep Learning Approach for Business Process Event Prediction”. In: *2017 IEEE 19th Conference on Business Informatics (CBI)*. Vol. 01. 2017, pp. 119–128. DOI: 10.1109/CBI.2017.46.