A REPORT

ON

"Measuring user engagements using Activity sequence Embeddings"

BY

NAME: -

USNEEK SINGH

ID NO: -

2019A7PS0127P

AT

Happiest Minds Technologies, Bangalore

A Practice School-I Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

JUNE, 2021

A REPORT

ON

"Measuring user engagements using Activity sequence Embeddings"

BY

NAME: -	ID NO: -	Discipline: -
USNEEK SINGH	2019A7PS0127P	Computer Science

Prepared in partial fulfilment of the

Practice School-I Course Nos.

BITS F221

AT

Happiest Minds Technologies, Bangalore

A Practice School-I Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

June,2021

ACKNOWLEDGEMENTS

I am obliged to practice school division of Bits Pilani for providing me with the opportunity to work at Happiest Minds Technologies. My sincere thanks to Happiest Minds Technologies for providing me the project. I would like to extend my thanks to Mr. Umesh Menon for guiding me on the project. And my special thanks to Dr. Ramakrishna Dantu for coordinating the entire process of practice school and for his constant guidance and supervision.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

(RAJASTHAN)

Practice School Division

Station: Happiest Minds Technologies **Location:** Bangalore

Duration: 7 weeks **Date of start:** June 1,2021

Date of submission: July 22, 2021

Title of the Project: Measuring user engagements using Activity sequence Embeddings

Student:

2019A7PS0127P, Usneek Singh, Computer Science (CS)

Name and designation of experts:

Mr. Umesh Menon, Principal Data Scientist, Happiest minds Technologies

Keywords: Embeddings, SGT, Autoencoders, LSTM, Attributed Sequences

Project Area: Deep Learning

Abstract:

This report covers the various techniques applied to clickstream data collected from a client website to generate embeddings which contain information about the features learnt from the embeddings. Various operations were performed using the pandas library to generate sequences from the log entries. Three different techniques; - SGT (Sequence Graph Transform), Autoencoders and attributed sequence were applied. The evaluation was done by training the classifier on train set based on those embeddings to predict output (binary classification). Finally predicted values were compared with actual values on test set to determine accuracy and f1 score.

Contents

1 Introduction

In this world of technology its easy to observe a person's actions on internet and collect data about it. The data collected can be used to analyse person's behaviour and can be useful in recommendations, determine outputs, etc. A sequence of activities that a person follows has a lot to say about that person. We have observed that similar sequences can be used to extract features from it called embeddings. In this report we talk about such similar embeddings, what are the ways to extract them and how can we use them in different aspects.

What are embeddings?

Embeddings are feature vectors containing different numerical values which we can say represent an individual feature. In simple words, an embedding vector for an object like orange would have higher value for the feature that contains information if this object is to be eaten whereas embedding for king would have higher value in the dimension having information if the object is royal. The objects which are similar would have more similar embeddings like apple and orange whereas the embeddings for an orange and smartphone would be quite different. Suppose an embedding looks like the vector: [to be _eaten, Royal, Real _object]. So, for an object like orange embedding would look like [0.97, 0.04, 0.08] and for king it would look like [0.03,0.99,0.9]. And if we visualise these vectors in 2-D space we can see the vectors for more closely related objects would be closer on graph.

Why are embeddings useful?

Let's take a simple example from natural language processing to demonstrate importance of embedding vectors where they are quite popular. If I have taught my machine the sentence that "I would drink orange juice". So, I would also like it to understand that "I would drink apple juice" is also a correct sentence. But that to happen we need to machine to understand that apple and orange are alike. That's when embeddings come handy. Similarly, we can extend the use of embeddings in learning features of sequences. The similar sequences will have more closely related features and hence more close embeddings.

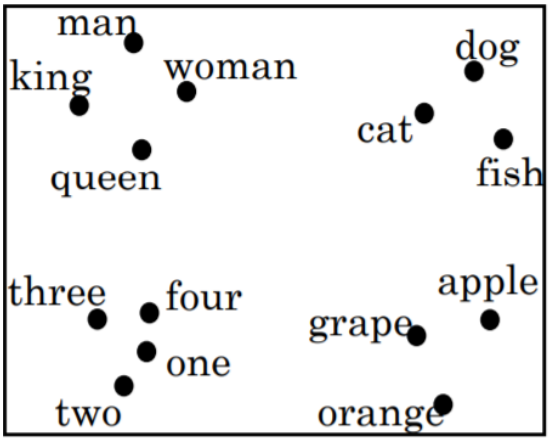

Fig 1: - Representation of embeddings of different words on a 2-D graph

There are some standard ways of converting words to embeddings like skip-gram model, negative random sampling, etc. For the purpose of sequence embeddings, I implemented three ways; -

1. SGT (Sequence Graph Transformation)
2. Autoencoder embeddings
3. Attributed sequence embeddings

2 Data

The data was obtained from an educational website. It contained log entries for each module of the website visited by user along with date of start and end. It also had information about working hours, grades obtained and type of activity done on that date. Firstly, we converted the data into pandas data frame. Now for each individual user we arranged the entries in ascending order of starting date and extracted the module name and arranged the items in form of a list. At the same time, we made another data frame which removes those users who have incomplete information registered i.e., they had no information whether they passed or failed in the course. We joined both frames and finally had the sequences for each user and final result (pass/fail). We will use this sequence data to create embeddings. The pandas library came in handy to get the data in desired form.

We will discuss in detail the various methods to make sequence embeddings.

3 Approaches to create embeddings

The three ways that we used were: -

3.1 Sequence Graph Transform (SGT)

Sequence Graph Transform (SGT), a feature embedding function, that can extract varying amount of short- to long-term dependencies without increasing the computation proposed. SGT embedding is a nonlinear transform of the inter-symbol distances in a sequence. Its name is attributed to the graphical interpretation of the embedding which shows the "association" between sequence symbols.

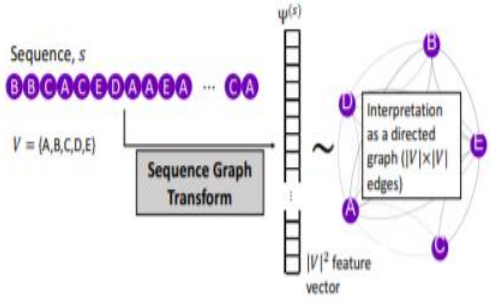

Fig 2: A diagrammatic representation of sequence graph transform Src:[1]

Let's consider a sequence ABCABCDAB, in this sequence we can extract sequence features in form of associations between the alphabets. Relative positions of two events in the sequence are considered for bringing in feature vectors.

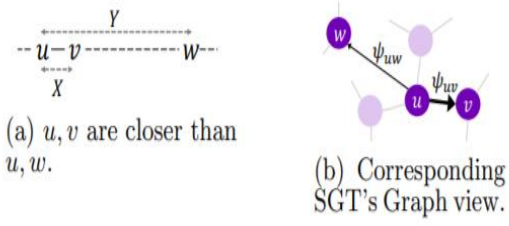

Fig 3: - More closely placed events have higher associations between them when converted in SGT graph. Src:[1]

Let's consider the practical approach to apply SGT

We need to install SGT library and import SGT class. This class has various parameters associated with it like: -

1. Kappa- Tuning parameter to decide whether to extract long term dependency. Higher the value, lesser the long-term dependency captured in the embedding. Common values used are- {1,5,10}.
2. Lengthsensitive- This is set true if embedding should have information about the length of the sequence.

3. Flatten -It is set true if we want single dimension embedding to be returned. By default, we gat 2-D embedding.
4. Mode – It can be set as default or multiprocessing depending upon the requirements of processors required for the embedding extraction.

Corpus is list of sequences and we can use the transform() method to convert the corpus into embedding vectors.

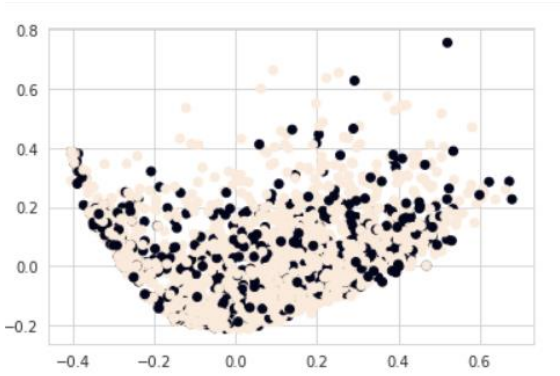

Fig 4: It shows the reduced dimension visualisation of embeddings using PCA component analysis.

	(M1, M1)	(M1, M10)	(M1, M11)	(M1, M12)	(M1, M13)	(M1, M14)	(M1, M15)	(M1, M16)	(M1, M17)	(M1, M2)	(M1, M3)	(M1, M4)	(M1, M5)	(M1, M6)	(M1, M7)	(M1, M
id																
1419982.0	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000e+
985076.0	0.367879	0.000000e+00	3.405135e-02	0.000000	0.003152	0.000000	0.000624	0.000000	0.000058	0.000000	1.147234e-01	0.000000	0.000000e+00	7.812614e-06	0.000000	0.000000e+
1423013.0	0.000000	0.000000e+00	7.582560e-10	0.000000	0.000000	0.000000	0.000588	0.184334	0.000002	0.004807	1.907828e-10	0.000000	1.352636e-12	1.041639e-08	0.003369	2.092187e-
1228078.0	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000e+
178878.0	0.036788	8.569023e-08	7.174264e-06	0.000038	0.000038	0.000226	0.024900	0.009976	0.004038	0.022036	1.824193e-02	0.023143	2.664473e-02	1.393615e-04	0.007756	7.566968e-
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1116599.0	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	3.678794e-01	0.006738	4.978707e-02	1.353353e-01	0.007235	0.000000e+
1411918.0	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000e+
1321877.0	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000e+
1386226.0	0.049787	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.930975e-01	0.000000	0.000000e+00	3.678794e-01	0.135335	7.103662e-
1390793.0	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000e+

Fig 5: - The embedding extracted from our data. It's a snapshot for a few ID's

4.2Autoencoder Embeddings

Let's first try to understand what an autoencoder is. It has two parts: encoder and decoder. The task of this neural network is to reconstruct the sequence back from the input. Our task is to feed the entire sequence in the model. The encoder part tries to learn features from it which is fed into decoder part which tries to reconstruct the sequence from the features. Our embeddings would be the output of encoder part. So, our target is to train the entire model and extract the output of encoder. We need to learn about the layers in our model: -

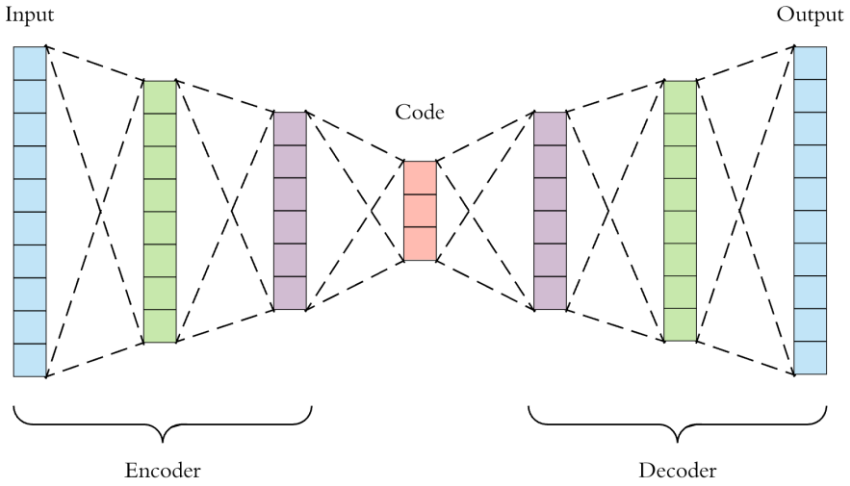

Fig 6: - The general model of an autoencoder

We implemented the sequence2vec model with Keras. The encoder is a single-layered bidirectional LSTM and the decoder is a single-layered unidirectional LSTM. By employing a bidirectional LSTM as the encoder, the sequnece2vec model is able to preserve information from both past and future. In our experiments, we trained the sequence2vec model by applying the Adam optimizer for 100 epochs. Once this is done, the trained encoder can be used to generate sensor embedding zi for each sequence si.

One important thing to be noted is that we need to tokenise the sequences before passing them as input. Tokenizer class in python maps each of our word in sequence to a number. It takes the most common words for a given input of n tokens. Then after tokenising the input and training the model, We can predict embedding vectors from encoder model.

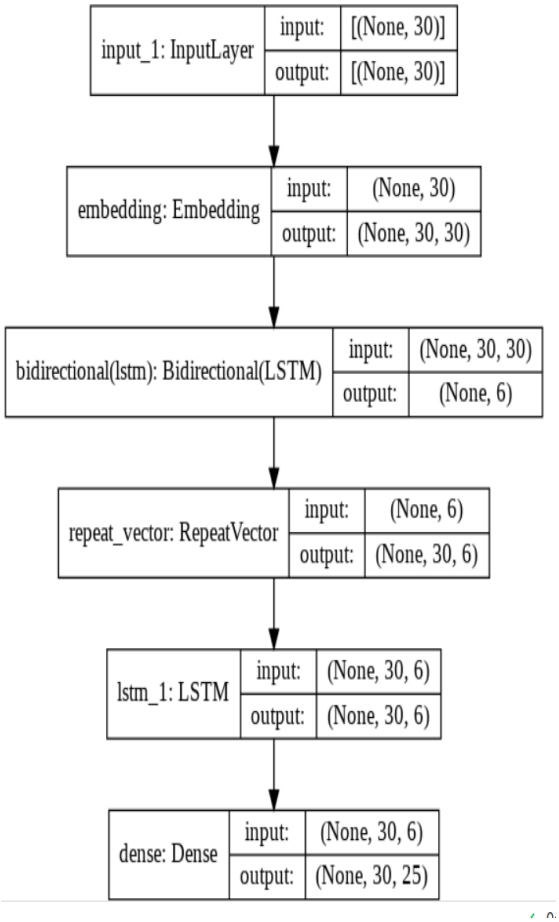

Fig 7 :- Model used in the autoencoder to generate embeddings.

3.3 Attributed Sequence Embeddings

Till now we have extracted our embeddings from sequences only. But there is also scope of extracting embeddings from sequences as well as attributes associated with those sequences linked together. Such embeddings should contain information which is able to represent attribute-attribute, sequence-sequence and attribute-sequence relations.

Let's us take an easy example to understand its meaning: -

In our data we have sequences containing information about each user's attempts on websites. And we also have information about the average score, time spent, type of activity, etc which can also be combined into attribute vectors. Now the combined vectors of both will be attributed sequence embeddings.

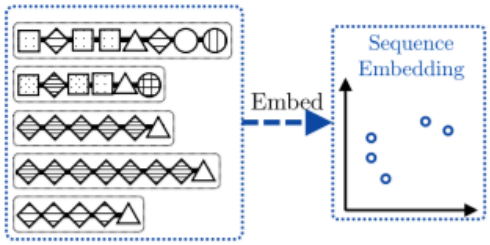

Fig 8: Sequence Embedding example Src:[3]

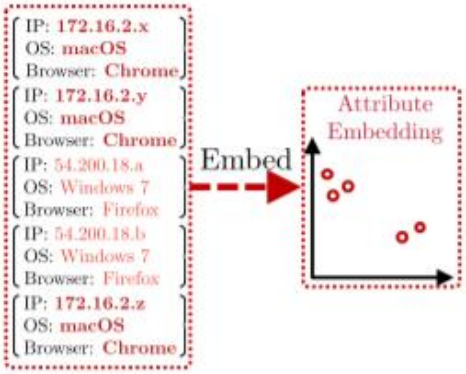

Fig 9: Attribute Embeddings example Src:[3]

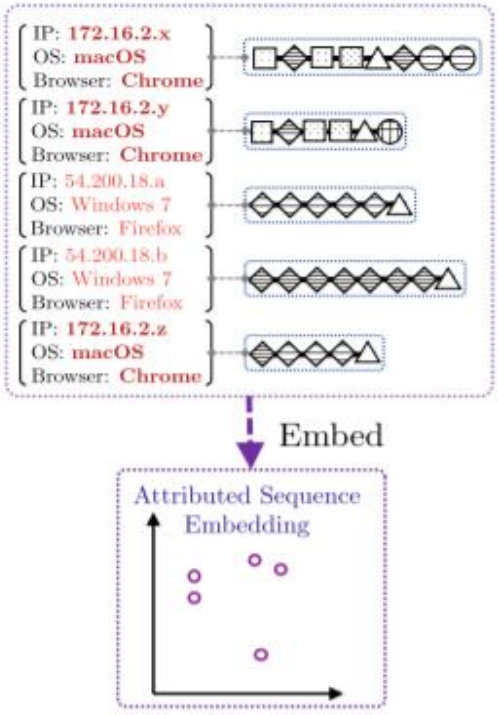

Fig 10: Attributed sequence Embedding example Src;[3]

The sequence network will have LSTM (Long short term memory) layers which conventionally works on following equations: -

$$
\begin{aligned}
\mathbf{i}_k^{(t)} &= \sigma\left(\mathbf{W}_i\boldsymbol{\alpha}_k^{(t)} + \mathbf{U}_i\mathbf{h}_k^{(t-1)} + \mathbf{b}_i\right)\\
\mathbf{f}_k^{(t)} &= \sigma\left(\mathbf{W}_f\boldsymbol{\alpha}_k^{(t)} + \mathbf{U}_f\mathbf{h}_k^{(t-1)} + \mathbf{b}_f\right)\\
\mathbf{o}_k^{(t)} &= \sigma\left(\mathbf{W}_o\boldsymbol{\alpha}_k^{(t)} + \mathbf{U}_o\mathbf{h}_k^{(t-1)} + \mathbf{b}_o\right)\\
\mathbf{g}_k^{(t)} &= \sigma\left(\mathbf{W}_g\boldsymbol{\alpha}_k^{(t)} + \mathbf{U}_g\mathbf{h}_k^{(t-1)} + \mathbf{b}_g\right)\\
\mathbf{c}_k^{(t)} &= \mathbf{f}_k^{(t)} \odot \mathbf{c}_k^{(t-1)} + \mathbf{i}_k^{(t)} \odot \mathbf{g}_k^{(t)}\\
\mathbf{h}_k^{(t)} &= \mathbf{o}_k^{(t)} \odot \tanh\left(\mathbf{c}_k^{(t)}\right)
\end{aligned}
$$

To accomplish this task, we will make a recurrent model consisting of many LSTM cells which proceeds in the following steps:

1. Made attribute vectors (three attributes used were: Score for Activity: [Study: 1, Quiz: 1.15, Test: 1.25], Total time spent on entire course and Average score during the course).

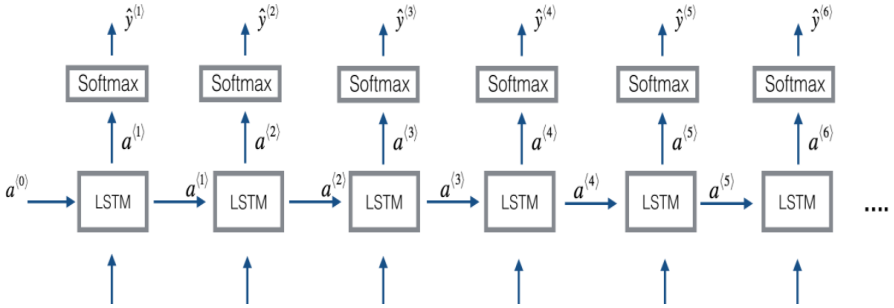

Fig 11: Diagrammatic representation of LSTM model used for created

2. Sequence vectors were padded to length 30 and fed into recurrent model which takes input at each time steps and tries to predict next sequence step. The task is to minimise the likelihood of incorrect prediction of next time event.

$$L_S = -\sum_{t=1}^{l_k} \boldsymbol{\alpha}_k^{(t)} \log \mathbf{y}_k^{(t)}$$

3. At the first-time step, hidden state is added with attribute vector before passing to the softmax classification. For this to happened dimension of LSTM unit be equal to length of attribute vector

$$\mathbf{y}_k^{(t)} = \delta\left(\mathbf{W}_y \mathbf{h}_k^{(t)} + \mathbf{b}_y\right)$$

$$\mathbf{h}_k^{(t)} = \mathbf{o}_k^{(t)} \odot \tanh\left(\mathbf{c}_k^{(t)}\right) + \mathbb{1}(t=1) \odot \mathbf{V}_k^{(M)}$$

The last cell state obtained from LSTM cell will be used as an embedding for the sequence

4 Evaluation

We used the embedding vectors to classify the sequences which would lead to pass (y=1) or fail (y=0). We trained a classifier in keras using our trained set and then predicted the result on test set to determine the accuracy with all three types of embeddings. Next section explains the classifier

4.1 Experiment

The experiment is very simple.

X(input)= Sequence embedding of user

Y(Output)= Pass/Fail

We split our total data into train,validation and test set in 3:1:1 ratio. We trained our classifier one by one with different embedding. Classifier model was simple with the following layers.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 64)                25664
_________________________________________________________________
activation_3 (Activation)    (None, 64)                0
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 32)                2080
_________________________________________________________________
activation_4 (Activation)    (None, 32)                0
_________________________________________________________________
dropout_3 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 33
_________________________________________________________________
activation_5 (Activation)    (None, 1)                 0
=================================================================
Total params: 27,777
Trainable params: 27,777
Non-trainable params: 0
_________________________________________________________________
```

Fig 12:- Summary of the classifier

4.2 Results

We will analyse our results on basis of two metrics: - accuracy and F1 score.

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

$$specificity = \frac{TN}{TN + FP}$$

Fig 13:- The definition of accuracy and F1 score

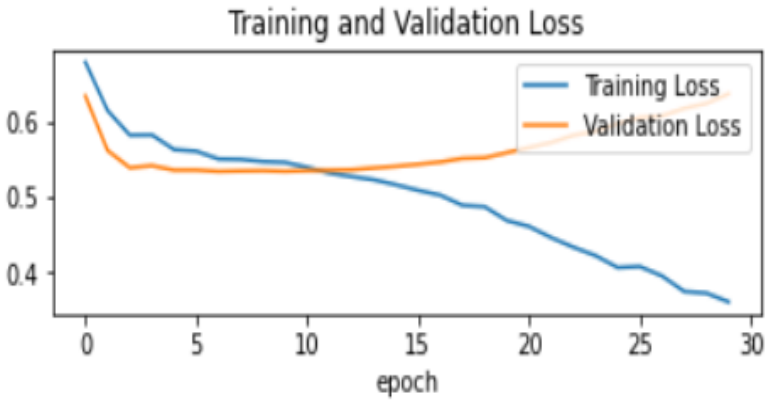

Fig 14:- Determing training and validation loss for SGT embeddings

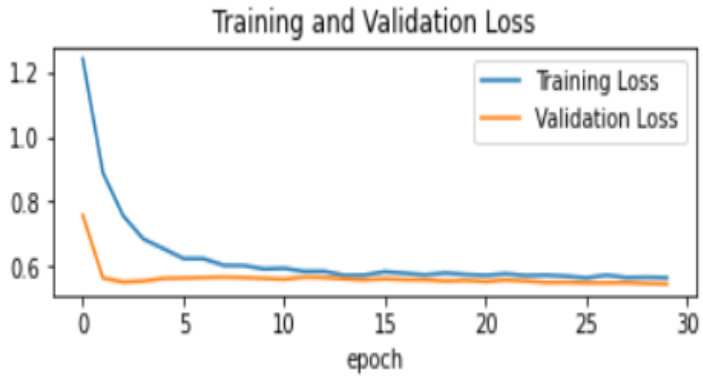

Fig 15:- Determing training and validation loss for Autoencoder embeddings

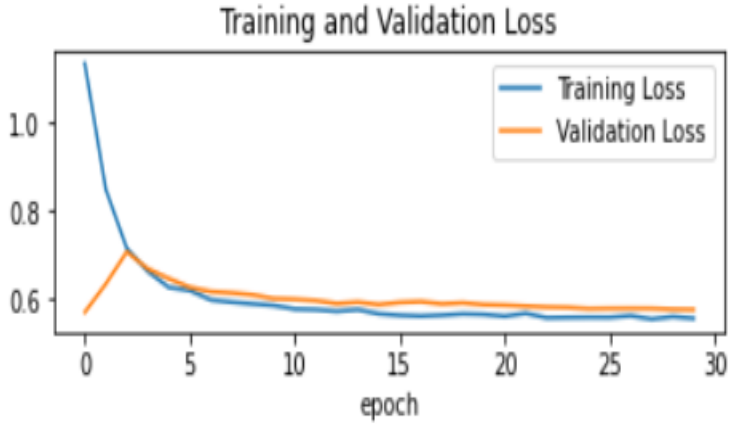

Fig 16: - Determining training and validation loss of Attributed Sequence embeddings

S.NO	Embedding Type	Training Accuracy	Test Accuracy	Test F1 score
1.	SGT	75.95%	76.4%	0.866
2.	Autoencoder	75.59%	76.43%	0.866
3.	Attributed sequences	76.53%	77.06%	0.870

Table 1: - Tabular results of all embedding types on classifier

Conclusion

All three types of embeddings give similar results which shows that all of them have equal potential in solving the problems of sequence embeddings. The training accuracy of SGT, autoencoders and attributed is 75.95%, 75,59%, 76.53% respectively. Whereas the test accuracy for SGT, autoencoders and attributed is 76.4%, 76.43%, 77.06% respectively. All embeddings are able to produce sufficiently good embeddings. The F1 score for all three embeddings is nearly same. Hence, we come to the conclusion that all three methods work nearly similar and able to work well in training the classifier.

5 Potential of the Study

The sequence embeddings have lot of potential in the future. They help to understand the actions of user. They help us to relate events by identifying the vents which always occur together. They can be used to separate the users which have peculiar sequences which has use in detection of network intrusion, etc. The related events in the sequence can be used

for recommendation purposes. Apart from these like we already used we can use these sequence embeddings for classification purposes. Although these embeddings are different from conventional word embeddings but they have a great potential ahead.

6 References

1. [1] Ranjan, Chitta & Ebrahimi, Samaneh & Paynabar, Kamran. (2016). Sequence Graph Transform (SGT): A Feature Extraction Function for Sequence Data Mining.
2. [2] Ghods, Alireza & Cook, Diane. (2019). Activity2Vec: Learning ADL Embeddings from Sensor Data with a Sequence-to-Sequence Model.
3. [3] Zhuang, Zhongfang et al. "Attributed Sequence Embedding." *2019 IEEE International Conference on Big Data (Big Data)* (2019): 1723-1728.
4. http://yaronvazana.com/2019/09/28/training-an-autoencoder-to-generate-text-embeddings/

7 Glossary

1. **pandas**: - pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.
2. **PCA**: - Principal component analysis (PCA) is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance.
3. **Neural network**- A neural network is a series of algorithms that endeavours to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature.
4. **LSTM**- Long short-term memory is an artificial recurrent neural network architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points, but also entire sequences of data.
5. **Adam optimiser**- Adaptive Moment Estimation is an algorithm for optimization technique for gradient descent. The method is really efficient when working with large problem involving a lot of data or parameters. It requires less memory and is efficient.
6. **Training, validation and test sets**: - The "training" data set is the general term for the samples used to create the model, while the "test" or "validation" data set is used to qualify performance. Perhaps traditionally the dataset used to evaluate the final model performance is called the "test set".
7. **Keras**: - Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.