Bayesian Sequential Recommendation for Mental Health Interventions

Submitted for the Master of Science,
London School of Economics, University of London

Department of Statistics 2021/2022

In Collaboration with

Koa Health

Candidate Numbers

39290, 38193, 31093

Supervised by

Dr. Marcos Barreto

August 24, 2022

Acknowledgements

We would like to express our deep appreciation to our supervisor, Marcos Barreto, for his patience and continuous support.

Also, we are super grateful to Bartek Skorulski, Gabriele Sottocornola, Federico Lucchesi, Ludovik Coba, and Giovanni Maffei (formerly Koa Health) from the Koa Health team for their inspiration and guidance. Especially, Gabriele, who has provided us with lots of support during the past few months.

Contents

List of Figures

List of Tables

Executive summary

This paper presents using sequential recommender system to improve mental health condition via mobile-based intervention. In the paper, recurrent neural networks (RNNs) and transformers are mainly applied. RNN and transformer models are widely used for recommender system in e-commerce and other user-centered platform. It recommends items, music or movies users might like by learning users' past preferences, or referring to other users with similar user features.

Since mental health problems have gained increasing attention all over the world, there was a growth in demand for mobile-based mental health applications. Users can follow some suggestions recommended by the application based on their mental health condition. Such application can effectively alleviate mental health problems such as stress, anxious and low in confidence. However, how to effectively learn from users' past actions and recommend suitable activities for them has not been extensively researched.

In order to fill this void, we applied some possible deep learning models using the offline dataset from Koa Health's Foundations app, and evaluated models using the latest user activities in an offline environment. Our results indicate it is possible to effectively learn from users' past activity patterns and give out suitable recommendation. By using dropout approximation, we were also able to obtain distributions of users' preference in all activities.

We first researched on related work and evaluated their strength and weakness. Then we provided a theoretical explanation of the deep learning models and evaluation methods we applied, as well as the configuration in the experiments. The experiment results indicate that our models can effectively reach the goal.

Datasets

Two datasets with similar attributes were used side by side in our experiments. The Koa health dataset consists of each user's clicked activity history collected from Foundations app. The MovieLens dataset is a larger and more balanced benchmark dataset for recommender

system. For each user, we only kept the last 11 clicked activities, with the first 10 activities used as input to the models, and the last activity as target.

Methods

We introduced three embedding generation methods that can be classified as either self-trained or pre-trained. The Act2Vec embedding was trained using clicked activity/movie sequences from two proposed datasets. On the other hand, two pre-trained models, namely Doc2Vec and SBERT were applied to each activity/movie titles. For Act2Vec, the similarity between each activity/movie was obtained based on the context information of such activity, while for the pre-trained models, the embedding vectors were obtained based on the title level information learned on larger text corpus.

To better capture the sequential relationship in the inputs, Bi-LSTM model and transformer are proposed, with additional regularisation methods to reduce overfitting during training. Besides, a feedforward model with one fully-connected hidden layer was used as a non-sequential comparison. In addition, a popular activity recommender system was proposed as a non-personalised baseline. In order to obtain a distribution for each prediction, dropouts were implemented during inference.

Offline evaluation

We introduced four evaluation metrics for offline evaluation: recall, soft evaluation, coverage and average variance. The recalls between each model were compared at 1,2 and 5 correspondingly. Soft evaluation metric provided a less strict assessment by evaluating whether the prediction falls into the same focus areas/genres with ground truth. Coverage rates were included in order to avoid the situation where a model repeatedly recommend one activity/movie. In order to evaluate model uncertainty, average variance was used as an additional metric to quantify uncertainty.

Contributions

On Koa data, our Bi-LSTM model with pre-trained Doc2Vec embedding hits $9.06\%, 13.13\%$, 22.26% on recall@1,2,5 respectively with diversified predictions, while obtaining 36.12% on

soft evaluation (AreaScore). We believe this is a remarkable result with the data we had from Koa's side. On MovieLens dataset, our Bi-LSTM and transformer models both achieved the state-of-the-art results, where transformer model with Doc2Vec embedding leads to $11.57\%, 16.86\%, 26.41\%$ on recall@1, recall@2 and recall@5 respectively. It also maintains good coverage property at over 70%, while keeping the average model variance low at decimal level. These results imply the feasibility of training and testing sequential recommendation engines under an offline manner.

Our sequential based recommendation systems can help improve recommendation performances, and since our models performed better on MovieLens dataset, we believe if more data comes from Koa, better results will be obtained. Furthermore, our recommendation engine can now be embedded into Koa's multi-armed bandits framework where it replaces Gaussian process to improve the performance. We believe our models are helpful on providing users with more suitable and more personalised interventions to enhance their mental health.

Table of notations

Notation	Description	Notation	Description
$\boldsymbol{C}$	Memory cell	S	Test set
$\widetilde{\boldsymbol{C}}$	Candidate Memory cell	$\boldsymbol{a}$	Attention vector
$\boldsymbol{F}$	Forget gate	$\boldsymbol{b}$	Bias in neural network
$\boldsymbol{H}$	Hidden state	$\boldsymbol{c}$	Context vector
$\boldsymbol{I}$	Input gate	$\boldsymbol{f}$	Focus area vector
$\boldsymbol{K}$	Keys	$\boldsymbol{l}$	Activity/movie title
$\boldsymbol{O}$	Output gate	$\boldsymbol{t}$	Paragraph token
$\boldsymbol{Q}$	Queries	$\boldsymbol{w}$	Word
$\boldsymbol{U}$	Recurrent weight matrix	d	Dimension of embedding
$\boldsymbol{V}$	Values	h	Attention head
$\boldsymbol{W}$	Weight matrix	m	Number of attention head
$\boldsymbol{X}$	Input matrix	r	Dropout rate
$\boldsymbol{Z}$	Multi-head attention output	y	Ground truth target
E	Set of categories that y belongs to	z	Normalized input
I	List of all activities/movies	α	Scalar value
$\mathcal{N}$	Euclidean norm	β	Bias
K	Number of classes	λ	Control parameter for regularization
L	Loss function	μ	Mean
R	List of top-k recommended activities/movies	σ^2	Variance

1 Introduction

1.1 Mental Health Condition and Possible Interventions

Mental health has now become a worldwide problem. The World Health Organization(WHO) (2019) initiative points out that more than 80% of people are experiencing mental health conditions, and approximately 70% of these people are employed (Attridge, 2019). Some commonly used interventions in clinics are Mindfulness-based cognitive therapy, Solution-focused brief therapy, Interpersonal psychotherapy, etc (Kuyken et al., 2016). However, such interventions can be time-consuming and less cost-effective. Younger generations and working professionals in the network era tend to employ more accessible and less time-consuming interventions when having slight mental health issues. In order to meet this growing demand, several mobile phone-based applications aiming to ameliorate such stressed workplace mental health circumstances. Some studies have shown that such novel interventions for mental health have potential influence on users' mental health (Goldberg et al., 2022), especially in areas such as anxiety, depression and stress. Also, it shows that mobile interventions is a vibe method for encouraging participants' engagement via self-evaluation and self-monitoring (Fuller-Tyszkiewicz et al., 2018).

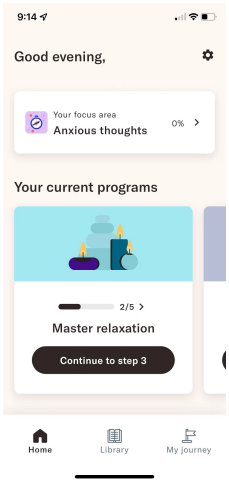

Figure 1: Dashboard of Foundations app.

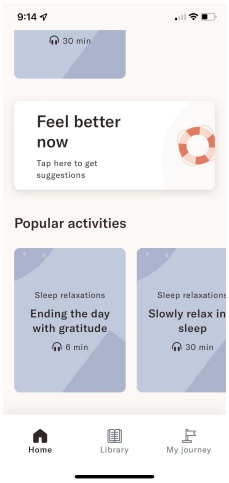

Figure 2: Personalised popular activities recommendation.

Foundations app is such an application developed by Koa Health to support mental wellbeing. The interface is shown in Figure 1. The purpose of the application is to understand how stress impacts our lives and find accessible solutions to improve wellbeing. It includes a library of content grounded in science to help users deal with stress, poor sleep, worries, anxious thoughts, low self-esteem and more. Users can use the wellbeing check-in activity to keep an eye on their mental state on a daily basis. Koa Health has also developed a recommender system to provide personalised recommendations to users in order to increase user engagement.

1.2 Recommender System

Recommender system is a class of software tools and techniques that offers suggestions to users by providing one or a list of selected items (Ricci et al., 2011). In recent years, recommender systems have become increasingly important due to the boom in e-commerce and online platforms.

The underlying architectures of recommender systems can vary to a great extent according to different objectives. For example, three types of recommender systems have been applied in Foundations app according to different needs and objectives. A recommender system based on item similarity is used to predict the next activity, given the one the user has just completed. A rule-based recommender system is used to generate a list of activities for a user after a well-being assessment, this recommender system is based on an expert-defined set of rules for providing suitable content. A data-driven recommender system based on Machine Learning/Artificial Intelligence models is employed to retrieve a list of personalised popular activities for each individual user. Figure 2 shows the interface of such personalised popular activities recommendation, and our research will be focused on such particular recommender system.

1.3 Target of the Project

Koa Health employs Gaussian process (Williams and Rasmussen, 2006) to determine the distribution of users' responses towards a particular activity in the Foundations app. A group of random variables are modeled as a multivariate normal distribution. There is a "range" of possible values for unobserved points when modeling the distributions between observations. This method allows the application to learn from users' historical activity sequences and find users who have similar activity history. The application then divides up each activity among all users and advertise the one with the highest reward.

Gaussian processes can be used to assess an activity's potential reward for a particular user as well as estimating the level of uncertainty. Thompson sampling (Thompson, 1933) then samples from the estimated distribution to balance the exploration-exploitation trade-off. The reward model will have a larger variance at the start of the exploration because of few observations, producing natural variability in the sampling process. The sampling process stabilises as the model becomes more dependable (lower variance), and the focus shifts to exploitation.

In our research, we replaced Gaussian process by a Bayesian neural network to estimate the reward distribution of a user response towards an activity, which measures both the mean and the variance of the user's action. Neural networks outperform shallow machine learning methods in most applications with high dimensional data, and are generally useful in the area involving texts and images (LeCun et al., 2015). Comparing to Gaussian process, neural network can be applied to more dataset scheme. Especially, recurrent neural networks (RNNs) (Rumelhart et al., 1986) have the ability to capture sequential user behaviour pattern. Hence, it will be more suitable in the scenario of a personalised recommender system. In order to achieve a similar effect to the Gaussian process, dropouts will be applied to achieve the uncertainty estimation.

Our purpose is to build a deep learning model that gives a personalised probabilistic estima-

tion of each popular activities, with a satisfactory accuracy and coverage rate. At the same time, this model should have the ability to obtain a distribution for each user and each activity at a given moment, providing a sampling pool for Thompson sampling to balance between exploration and exploitation trade-off. Our models have surpassed some existing models in terms of recall rate, as well as having a satisfactory coverage rate. At the same time, our models are able to capture uncertainty due to the implementation of dropout approximation.

This paper is structured as follows. Section 2 surveys the relevant literature regarding mobile-based mental health intervention, recommender system and model uncertainty evaluation. Section 3 provides an overview of the data used in the project, data pre-processing procedure and proposed methodology. Section 4 provides interpretation of the results on offline datasets. Section 5 concludes with some discussions and limitations. Finally, Section 6 discusses about some possible further work.

2 Literature Review

2.1 Mobile-based mental health intervention

As smartphone ownership rapidly grew through the last decade, Mobile Health (mHealth) has become a common option for monitoring patients and providing health support to the general public (Park, 2016). In particular, mHealth has been widely applied in mental health treatments and interventions. In 2015, WHO revealed that nearly 30% of mHealth apps being surveyed are related to mental health treatments and support (Chandrashekar, 2018). Studies have shown that mHealth applications have been proven to be effective in alleviating common mental health problems, such as depression and anxiety (Chandrashekar, 2018; Goldberg et al., 2022).

Mobile-based mental health intervention has several advantages compared to traditional interventions. First, it is more scalable and accessible to the general public due to its mobile-based nature. This can potentially reduce the burden on healthcare workers (Park, 2016). Second, since the affordability of mental health treatment can be an issue for low-income individuals (Feinson and Popper, 1995) and working professionals with compact schedules, such an intervention can be more cost effective. These advantages explain the growth in popularity of mental health apps over the past decade.

Despite these advantages, several challenges regarding such an intervention should also be considered. In particular, increasing user engagement remains a huge challenge for mental health app developers. For patients with mild depression symptoms, their activity engagement rate can be lower due to low self-motivations (Molloy and Anderson, 2021). User engagement rate can be improved by creating better contents, and recommending more personalised contents to each user. In this context, having a recommender system that recommends personalised popular activities can potentially increase user engagement rate, and achieve greater efficacy.

2.2 Recommender systems

Recommender system is designed to filter from a large pool of information, and provide suggestions to users with selected items based on several users' features (Ricci et al., 2011). The purpose of a recommender system is to retrieve a list of items that cater to the user's interest from a large set of data (Lu et al., 2015), and subsequently increase user engagement and retention rate.

2.2.1 General recommender systems

Collaborative filtering is one of the powerful personalisation tools driving the adaptive websites (Schafer et al., 2007). It is a technique that performs item selection based on users' preferences. By bringing the opinions of several relevant and huge online communities together, Collaborative filtering technology is able to filter from a large volume of data.

Another popular class of technique is the content-based filtering (Pazzani and Billsus, 2007). According to Pazzani and Billsus (2007), unlike collaborative filtering, content-based filtering makes recommendations based on the users' own preferences. However, such a traditional algorithm experiences scaling issues when it encounters large volume of data, consequently lower its performance. It is also proven to be troublesome when dealing with sparse data (Skovhoj, 2022).

2.2.2 Deep learning based recommender systems

In recent years, deep learning-based solutions are being pursued (Zhang et al., 2019). Deep learning models have advantages in dealing with recommender system scenarios because of the following characteristics: nonlinear transformation, representing learning, sequence modelling and flexibility. In 2015, Wang's team proposed one of the first attempts to employ deep learning approaches in the field of recommender systems, using Stacked Denoising Autoen-

coders to simultaneously learn collaborative and content-based features. This experiment was then put into a typical collaborative filtering model (Wang et al., 2015). Likewise, in 2016, Shen's team introduced a method using convolutional neural networks (CNNs) to extract item characteristics from text information of learning resources, such as the introduction and content of learning material (Shen et al., 2016).

All the algorithms mentioned are content-based modeling, which relies on explicit feedback from users. But in most cases, users' ratings towards items, or users' explicit feedback are hard to collect and quantify. Despite content-based modeling, Wu et al. (2016) created a session-based recommendation algorithm for a real-world e-commerce website. It uses simple RNNs to estimate what a user would buy next based on click history. RNNs are also applied in sequential models. Wu et al. applied the RNN model by incorporating text reviews and ratings at the same time. Unlike other text review-enhanced recommendation models, this one uses a character-level Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) network with user and object latent states to create reviews. Atassi et al. (2018) also proposed that the session-based Gated recurrent unit (GRU), which can be viewed as a simplified version of LSTM, outperformed content-based CNNs in the application of Arabic language program.

In 2019, the search and recommendation group of Alibaba, China's largest e-commerce website, proposed a transformer model to gather the underlying signals that drive users' activity patterns, and provide recommendations (Chen et al., 2019). The key advantage of such model is its ability to detect the dependencies among words because of the self-attention mechanism. In past two years, transformers has been found to have a better performance in both Nature language processing (NLP) tasks and recommender systems (Yadav and Singh, 2020; Liu et al., 2021).

2.3 Model uncertainty estimation in deep learning

Although deep learning has achieved tremendous success during the past decade (Ovadia et al., 2019), deep learning model itself lacks a representation of uncertainty, and often results in miscalibrated predictions (Ovadia et al., 2019; Maddox et al., 2019). In practical applications such as self-driving systems and medical diagnostics, knowing the corresponding confidence score for each prediction is crucial for decision making (Kendall and Gal, 2017). In the context of reinforcement learning, being able to obtain a confidence interval for each prediction is required for carrying out exploration vs exploitation tradeoff in Thompson sampling (Gal and Ghahramani, 2016).

In order to conquer this limitation, several approaches to estimating model uncertainty have been developed. Guo et al. (2017) proposed temperature scaling, a method based on Platt Logistic Scaling (Platt et al., 1999), to obtain a probability of confidence for the prediction results in classification tasks. The idea is to divide the logits obtained from the output layer by a hyperparameter T, and tune the value of T on a separate validation dataset to minimise the negative log likelihood. This method returns a confidence score which allows a more straightforward calculation of expected calibration error (Naeini et al., 2015). However, it does not return the mean and variance of the distribution.

Gal and Ghahramani (2016) proposed using Monte Carlo dropout to estimate uncertainty. This method extends model uncertainty estimation to both classification and regression models. Conventionally, the dropout layer is used as a regularisation technique for preventing overfitting(Hinton et al., 2012). Under this method, dropout layers in the model are set to training mode during both training and inference, and the resulting neural network is proven to be the approximated version of a Deep Gaussian process (Damianou and Lawrence, 2013). By setting different hidden units to zero using dropout, the prediction result differs each time and this allows the calculation of the mean and variance of the distribution. This method comes with a moderate computational cost of $O(n)$ during testing only.

Lakshminarayanan et al. (2017) introduced deep ensembles as a method of quantifying uncertainty. Under this method, the same neural network is repeatedly trained by setting different random seeds each time, and the outputs are obtained to calculate the mean and the variance of the distribution. This method has proven to produce better uncertainty estimates without the need of parameter tuning. It works especially well on out-of-distribution samples, since randomised initialisation setting allows it to explore a wider range of modes (Fort et al., 2019). However, deep ensembles come with a computational complexity of $O(n)$ during both training and testing, therefore, its application to more complex neural networks is restricted.

3 Proposed Methodology

3.1 Datasets Introduction

To obtain more comprehensive results, we trained and tested our models using two datasets: one is the Koa dataset provided by Koa Health, and another is the larger MovieLens dataset from GroupLens research group (*Movielens 20m dataset*, 2021).

Koa Data The Koa dataset consists of Foundations app data and focus area data. The Foundations app data contains user activity information from 14/06/2021 to 14/06/2022. The schema is shown as below:

Attributes	**Explanation**
user Id.	the index of the user in one activity history
activity Id.	the index of the activity
activity title	the title of the activity
event	the type of user event in the activity history, either "impression" or "tap"
created at	the timestamp of the event being created
date	the date of the event being created

Table 1: Koa dataset Scheme.

Since we treated each event as the implicit feedback from users, we only kept the records with "tap" in event. After dropping the "impression" rows and the data processing steps, there were 130 unique activities left in the dataset. These activities are equivalent to the vocabulary set in NLP tasks.

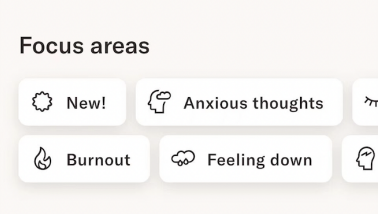

Figure 3: Focus area in Foundation app.

The focus area dataset is an activity level dataset that tells us which category each activity belongs to. There are 11 focus areas in total, and users can find activities by focus area in the application as Figure 3 presented. An activity can belong to more than one focus areas. During data pre-processing, a vector of length 11 was created for each activity in the vocabulary set according to its focus area. If an activity belongs to the i-th focus area, then the i-th element of the vector will be 1, otherwise -1.

MovieLens MovieLens is a widely used benchmark dataset for evaluating recommender systems. We used the version "MovieLens-20M" that includes 20 million user ratings. The datasets consists of rating data and movie data. The rating data was collected between 09/01/1995 and 31/03/2015.

Rating is usually considered as explicit feedback from users. In this case, we only filtered ratings over "3 stars" for each user. This procedure plays the same role as filtering "tap" in Koa data. In order to be consistent with Koa data, only the most popular 130 movies were selected according to its frequency. The schema of the datasets is shown as below:

Attributes	Explanation
userId	The user Id. of the record
movieId	The movie Id. of the record
rating	Made on a 5-star scale
timestamp	Seconds since UTC

Table 2: Rating Dataset Schema

Attributes	Explanation
movieId	The movie Id.
title	The title of the movie
genres	A list of genres of the movie

Table 3: Movie Dataset Schema

There are 18 genres in the movie dataset, and a vector representing the genres of the movie was generated. These vectors are equivalent to the focus area vectors in Koa dataset. We then merged rating data and movie data on movieId. After that, we only kept the rows that were created after 2010/01/01.

3.1.1 Data Pre-processing

For both the Koa and MovieLens dataset, we propose similar data pre-processing procedure. First, all attributes mentioned earlier were grouped by the same user. For Koa data, each user record contains a sequence of activity titles and a sequence of corresponding focus area vectors of those titles. Both sequences were ordered by ascending time. For the activity sequence, if a user has consecutively clicked on the same activity, we only kept the last one of that activity. Then for each user, we added a sequence of time differences between neighboring activities. We then dropped the rows that only contained one activity.

Data statistics are shown in Table 4. It can be inferred that Koa dataset has fewer actions per user (on average). Even though both dataset have 130 activities, the number of users in MovieLens dataset is nearly 200 times more than the number of users in Koa dataset.

Dataset	Number of users	Number of activities	avg. activities/user	Time span (years)
Koa	671	130	4.21	1.0
MovieLens	132179	130	28.95	5.25

Table 4: Dataset Statistic (after preprocess).

3.1.2 Prepare for models

The purpose of the model is to learn the user's history activity behaviour and predict the most possible activity that the user would like to click next. Hence the last activity in each user sequence was used as the target.

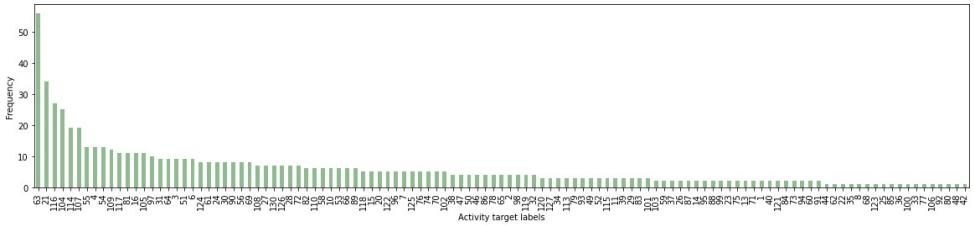

Figure 4: Activity targets distribution

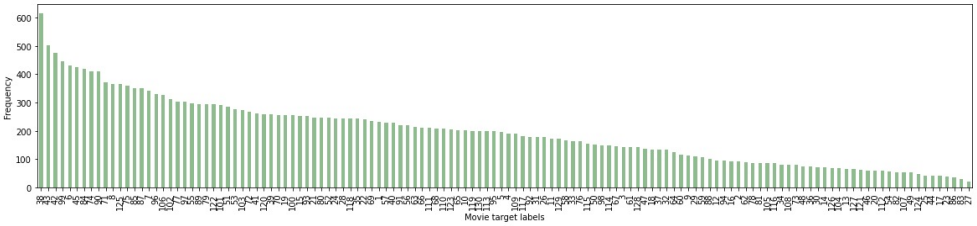

Figure 5: Movie targets distribution

By looking at the long-tail distribution of activity targets in Figure 4, we can tell some activities in the target set play a dominant role, so the model trained on Koa data might be bias due to the data imbalance. On the other hand, the target set of movie dataset shown in Figure 5 is better balanced.

	user	input	time_dif	focus_area_combine	target
0	9895	[85, 90, 21, 39, 100, 120, 28, 5, 19]	[0.005, 0.004884259259259259, 0.00100694444444...	[[-1, -1, -1, -1, -1, -1, 1, -1, -1, -1, -1, -...	33
1	54241	[42, 72, 60]	[0.0004976851851851852, 0.0020486111111111113,...	[[1, -1, -1, -1, -1, 1, 1, -1, -1, -1, -1, -1,...	105
2	44672	[90, 102, 87, 101, 74]	[6.944444444444444e-05, 3.472222222222222e-05,...	[[-1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1,...	7
3	86138	[120, 67, 97, 5, 61, 51, 130, 105, 79, 19]	[0.00037037037037037035, 0.0018055555555555555...	[[-1, -1, -1, -1, 1, 1, 1, -1, -1, -1, -1, -1,...	34
4	96999	[52]	[5.787037037037037e-05]	[[-1, -1, -1, -1, -1, 1, 1, -1, -1, -1, -1, -1...	53

Figure 6: MovieLens data after preprocessing

Figure 6 shows the pre-processed MovieLens data. In order to balance the effectiveness and comprehensiveness of users' past activity information, we selected the latest 10 activities in the remaining sequence as the input for each user. For users who have clicked on less than 10 activities, we used a padding value of zero to make up for the insufficient activities. In order to eliminate the weight generated by padding values, we masked zero to guarantee accuracy during training. This is why -1 and 1 are used in focus area vectors instead of 0 and 1. If time differences were added as one of the inputs, they were normalised on a -1 to 1 scale using the normalisation layer.

3.2 Embeddings

Three different embedding generation methods were experimented. We propose Act2Vec, which has similar underlying logic to the Word2Vec (Williams et al., 2017) model. In our case, each acivity Id. was mapped to the embedding space instead of each word. On top of this, we propose using two pre-trained models to generate embeddings. This is based on the assumption that movie and activity titles can be viewed as paragraphs or sentences. Table 5 briefly summaries three types of embeddings used in our models.

Embedding name	**Pre-trained**	**Training data**	**Input**	**Output dimension**	
				Koa	**Movie-Lens**
Act2Vec	False	User clicked activity /movie Id. sequences	Activity /movie Ids	4	8
Doc2Vec	True	English Wikipedia DBOW and Associated Press News DBOW	Activity /movie titles	300	300
SBERT	True	Stanford Natural Language Inference (SNLI) (Mikolov et al., 2013) and Multi-Genre NLI (Bowman et al., 2015)	Activity /movie titles	768	768

Table 5: Embedding summary

3.2.1 Act2Vec Embeddings

Act2Vec embedding is obtained by training a Word2Vec neural network on sequences of user clicked activities/movie Ids. Each activity/movie Id is treated as a "word" $\boldsymbol{w}_i$ in the bag-of-words algorithm, and each activity sequence is a tokenized sentence consisting of at least one $\boldsymbol{w}_i$. During training, each $\boldsymbol{w}_i$ was used as the target word, and a set of neighbouring words $\{\boldsymbol{w}_{i-n}, ..., \boldsymbol{w}_{i-1}, \boldsymbol{w}_{i+1}, ..., \boldsymbol{w}_{i+n}\}$ was used as context words, where n was defined by the window size.

At the beginning, each $\boldsymbol{w}_i$ and its context word vectors are randomly initialised with a pre-defined dimension of d. During training, we used the Continuous Bag of Words (CBOW) model (Williams et al., 2017), where the sum of context word vectors were calculated, and this new vector was used to update $\boldsymbol{w}_i$. The neural network was trained such that the average log probability:

$$\frac{1}{K} \sum_{i=n}^{K-n} \log\, p(\boldsymbol{w}_i | \boldsymbol{w}_{i-n}, ..., \boldsymbol{w}_{i+n})$$

was maximised.

We trained the model for 5 epochs using Gensim Word2Vec model, with a sliding window of size 5. For each activity/movie in the vocabulary set, we map it to its corresponding updated $\boldsymbol{w}_i$ after training, and this $\boldsymbol{w}_i$ is used as the embedding. The dimension d was tuned such that the customised loss function was minimised, which will be further explained in 3.2.4.

Word2Vec models are often trained on large corpus of texts in order to obtain a desirable semantic similarity. However, such a goal was hard to achieve due to the limited size of Koa data. In order to tackle this problem, we applied pre-trained Doc2Vec (Le and Mikolov, 2014) and SBERT (Reimers and Gurevych, 2019) models on activity and movie titles.

3.2.2 Doc2Vec Embeddings

The Doc2Vec model was trained by Lau and Baldwin (2016) on Wikipedia articles and Press news text corpus, with a vocabulary size of 915,715 words. Unlike Word2Vec, Doc2Vec is a paragraph level embedding model. Given a paragraph $\boldsymbol{l}_i$ consisting of many words $\boldsymbol{w}_i$, the Doc2Vec obtains an embedding for each paragraph $\boldsymbol{l}_i$ by mapping each paragraph to a d-dimensional paragraph token $\boldsymbol{t}_i$. In the pre-trained model, the author used Distributed bag of words (DBOW) algorithm, which is similar to Skip-gram (Williams et al., 2017) in the Word2Vec model. During training, Each paragraph token $\boldsymbol{t}_i$ is used to predict $\boldsymbol{w}_i$ and its context words in this paragraph regardless of order. In this case, each $\boldsymbol{t}_i$ is unique between each paragraph, but each $\boldsymbol{w}_i$ is the same for the same word in different paragraphs. At the end of the training iteration, the updated paragraph token $\boldsymbol{t}_i$ is mapped to each paragraph $\boldsymbol{l}_i$ as its embedding, and each updated $\boldsymbol{w}_i$ can also be used as word embedding.

In our datasets, each activity/movie titles were treated as a paragraph. We used the pre-trained Doc2Vec model mentioned above to generate an embedding for each title. During prediction, the Doc2Vec model performed an inference step to obtain the new paragraph token, which is then used as the embedding for the corresponding activity/movie.

3.2.3 SBERT Embeddings

The SBERT model proposed by Reimers and Gurevych (2019) has a siamese network structure. Each of the two networks consist of a BERT encoding block followed by a pooling block. The output vectors of the two networks have the same dimension, and the dot product of the pair is calculated as a measure of similarity. The SBERT network was fine tuned by the authors using 1 million sentences. The author obtained a range of pre-trained models with output dimension varies between 300 and 1024, therefore, we used the same loss function as Act2Vec to obtain the best embeddings.

3.2.4 Embedding dimension tuning

To obtain the optimal embedding dimension for Act2Vec and SBERT, we defined a customised loss function L. For each embedding vector $\boldsymbol{v}_{i,i=1,\ldots,130}$ of dimension d in our vocabulary set, we first calculate the dot product similarity:

$$\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle_{i \neq j} = \sum_{l=1}^{d} \boldsymbol{v}_{il} \boldsymbol{v}_{jl}$$

and obtain a list of top 10 vectors $\boldsymbol{v}_{i_{top_k}}$ that has the largest dot product similarity with $\boldsymbol{v}_i$. We then calculate the sum of the Euclidean norms:

$$\mathcal{N}_i = \sum_{k=1}^{10} \| \boldsymbol{f}_i - \boldsymbol{f}_{i_{top_k}} \|_2$$

between $\boldsymbol{v}_i$ and its top 10 most similar embeddings, where $\boldsymbol{f}_n$ is the corresponding focus area vectors of $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i_{top_k}}$. The total sum of the Euclidean norms:

$$L = \sum_{i=1}^{130} \mathcal{N}_i \tag{1}$$

is used as an indicator of embedding quality. A relatively larger L indicates a larger difference between predicted embeddings and focus areas.

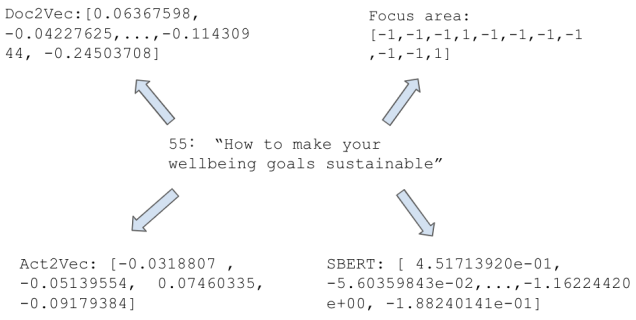

Figure 7: Embeddings and focus area of activity 55 in Koa dataset.

Embedding	Koa	MovieLens
Act2Vec	4397.51	4394.86
Doc2Vec	4786.84	4132.73
SBERT	4298.40	4444.44

Table 6: Embedding loss

Table 6 summaries the embedding losses calculated using the customised loss function L (1). The embedding dimension of Act2Vec has been tuned using a range of power of 2 values from 2 to 256, and the dimensions were chosen to be 4 and 8 for Koa and MovieLens respectively. SBERT dimension has been chosen from a range of avaliable values starting from 300, and our loss function indicated that 768 returns the lowest loss on both datasets. For Koa embeddings, SBERT returns the lowest loss and Doc2vec has the highest loss. For MovieLens dataset, Doc2vec embedding has the lowest loss.

3.3 Models

We propose three different models: Feedforward model, Bi-LSTM model and Transformer. The overall structures of models are shown in Figure 8:

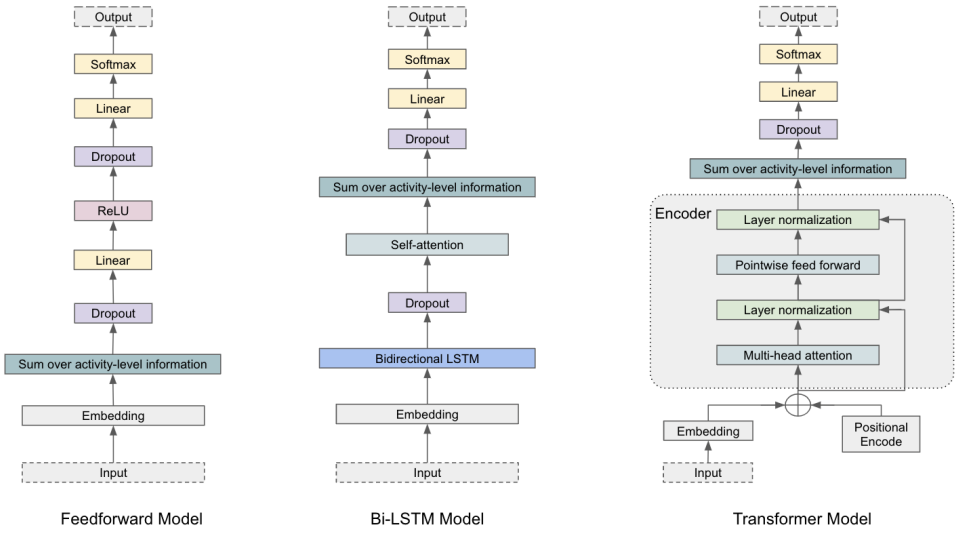

Figure 8: Model Structures

The models consist of two main parts: one is from the word-level to sentence-level, which was named by us as the sentence encoder. The other part is from the sentence-level to output-level, namely the output layer. A detailed explanation of the model structures are presented in the following sections.

3.3.1 LSTM Encoder

Traditional RNNs suffer from the vanishing gradient problem (Hochreiter, 1998) that often leads to short-term memory problems. In order to tackle this, several variants of RNNs have been developed over the years. Especially, LSTM has been widely used in processing sequential information. Figure 9 illustrate the detailed architecture of our LSTM model.

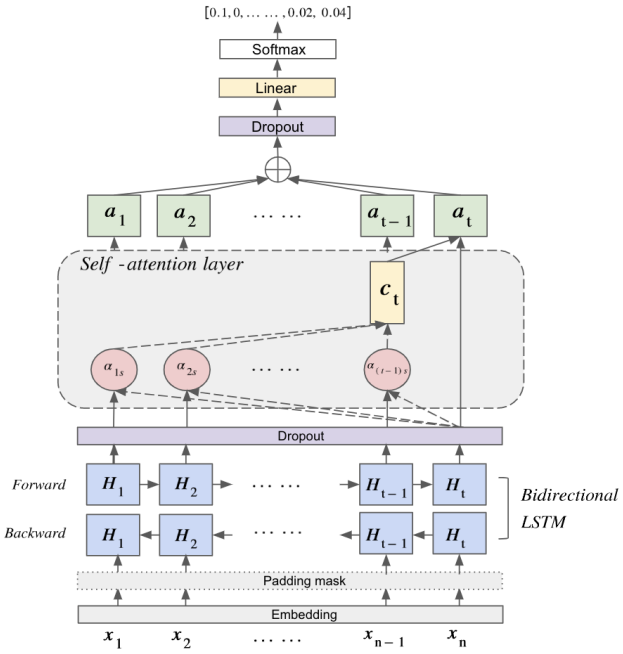

Figure 9: Bi-LSTM model structure.

Bidirectional LSTM LSTM (Hochreiter and Schmidhuber, 1997) added an input gate, an output gate and a forget gate on top of the cell in each RNN unit. The forget gate controls how much information to carry forward to the next hidden state (Gers et al., 2000). This

allows LSTM to deal with the vanishing gradient problem, and make predictions based on information obtained from longer time intervals. Given the input at the current timestamp $\boldsymbol{X}_t$, $\boldsymbol{W}$ and $\boldsymbol{U}$ are the weights for input $\boldsymbol{X}_t$ and output from last hidden state $\boldsymbol{H}_{t-1}$ correspondingly, and $\boldsymbol{b}$ defines the bias from the inputs. The input gate, forget gate and output gate of a LSTM cell are defined as the following:

$$\begin{cases} \boldsymbol{I}_t = \sigma(\boldsymbol{W}_i \boldsymbol{X}_t + \boldsymbol{U}_i \boldsymbol{H}_{t-1} + \boldsymbol{b}_i) \\ \boldsymbol{F}_t = \sigma(\boldsymbol{W}_f \boldsymbol{X}_t + \boldsymbol{U}_f \boldsymbol{H}_{t-1} + \boldsymbol{b}_f) \\ \boldsymbol{O}_t = \sigma(\boldsymbol{W}_o \boldsymbol{X}_t + \boldsymbol{U}_o \boldsymbol{H}_{t-1} + \boldsymbol{b}_o) \end{cases}$$

These three gates control the amount of information flowing into , going out from and being reset within the cell using a sigmoid function (σ). The memory cell and the candidate memory cell are defined as :

$$\begin{cases} \boldsymbol{C}_t = \boldsymbol{F}_t \boldsymbol{C}_{t-1} + \boldsymbol{I}_t \widetilde{\boldsymbol{C}}_t \\ \widetilde{\boldsymbol{C}}_t = tanh(\boldsymbol{W}_{\widetilde{c}} \boldsymbol{X}_t + \boldsymbol{U}_{\widetilde{c}} \boldsymbol{H}_{t-1} + \boldsymbol{b}_{\widetilde{c}}) \end{cases}$$

The two terms of $\boldsymbol{C}_t$ controls the amount of information to forget and to add on to the output.The hidden state and the true output from the LSTM cell is defined as:

$$\boldsymbol{H}_t = \boldsymbol{O}_t tanh(\boldsymbol{C}_t)$$

We used a Bidirectional LSTM (Bi-LSTM) layer (Schuster and Paliwal, 1997), which consists of two layers of LSTM cells in opposite directions. The output contains both information learned from past states and future states. In this case, the true output takes the form of a concatenated vector $\boldsymbol{H}_t = [\boldsymbol{H}_{t_{forw}}; \boldsymbol{H}_{t_{backw}}]$, and the output dimension is doubled by the input dimension. In our model, the Bi-LSTM has shown to have better performance than traditional LSTM by an average recall@2 of 2% under the same conditions.

Due to the sequential structure of LSTM, conventional regularisation methods such as dropout tend to perform poorly on LSTM (Turkin, 2017). To prevent overfitting, we applied elastic net regularisation on recurrent weight $\boldsymbol{U}$, known as the recurrent regularisation (Zaremba et al., 2014). The elastic net is a regularisation technique that is commonly used in linear regression tasks. It combines both the Lasso and Ridge penalties in the loss function of $\boldsymbol{U}$. The cost function can be written as

$$Cost = Loss + \lambda_1 \sum_{i=1}^{n} |\boldsymbol{U}_i| + \lambda_2 \sum_{i=1}^{n} \boldsymbol{U}_i^2$$

This provides a balance between $L1$ and $L2$ penalties when performing weight parameter selection, thus effectively alleviates the vanishing gradient problem. Figure 10 demonstrate the effect of recurrent regularisation on validation loss.

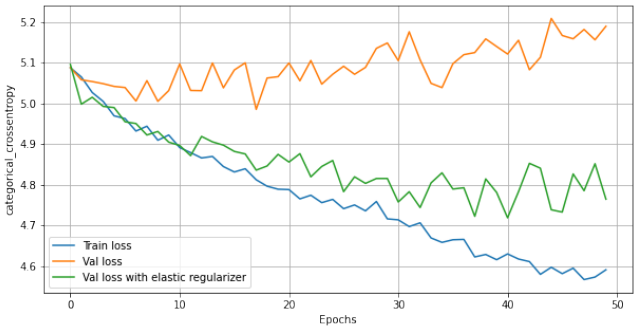

Figure 10: Bi-LSTM model trained on Koa data with Doc2Vec embedding.

Self-Attention A self-attention (Luong et al., 2015) layer was added after the Bi-LSTM layer. This allows the network to pay attention to certain parts of the input sequence. Given $\boldsymbol{H}_t$ to be the output from the Bi-LSTM layer, and $\boldsymbol{H}_{pre_s}$ to be the output from previous hidden states of $\boldsymbol{H}_t$, the self attention layer first computed the attention weight:

$$\alpha_{ts} = softmax(f(\boldsymbol{H}_t, \boldsymbol{H}_{pre_s})) = \frac{exp(f(\boldsymbol{H}_t, \boldsymbol{H}_{pre_s}))}{\sum_{i=1}^{s} exp(f(\boldsymbol{H}_t, \boldsymbol{H}_{pre_i}))}$$

where $f(\boldsymbol{H}_t, \boldsymbol{H}_{pre_i})$ is the attention score function. The calculation of the score function can be based on addition or concatenation. We chose the dot product attention score function since the computation is faster, which speeded up the training process. Therefore, the corresponding attention score function was $f(\boldsymbol{H}_t, \boldsymbol{H}_{pre_i}) = \boldsymbol{H}_t^T \boldsymbol{H}_{pre_i}$. We then obtained the context vector:

$$\boldsymbol{c}_t = \sum_{i=1}^{s} \alpha_{ts} \boldsymbol{H}_{pre_i}$$

where the output vectors from previous hidden states were weighted by the attention score. The output attention vector was then calculated by:

$$\boldsymbol{a}_t = tanh(\boldsymbol{W}_a[\boldsymbol{c}_t; \boldsymbol{H}_t])$$

To make sure the self-attention layer functions properly, a padding mask was added after the embedding layer so the self-attention layer would ignore padding value zero. In addition, the return sequence option in the Bi-LSTM layer was set to True so the outputs from previous hidden states would be returned. With the self-attention mechanism, the average recall metrics were boosted by over 30% than original Bi-LSTM models.

Dropout The dropout layer has two functionalities in the models. First, it serves as a regularisation technique. This is achieved by randomly selecting input units to be 0 with a proportion of r, where $r \in [0, 1)$ is known as the dropout rate. In our models, a range of dropout rates from 0.1 to 0.5 has been tuned, and 0.4 was chosen as it returned the highest recall on our validation sets.

Second, the dropout training mode is set to True in both the training part and the inference part, to achieve the effect of a Bayesian neural network. This is due to the randomness introduce by the dropout technique. Specifically, for each input sequence of each user, instead of returning a fixed prediction result, our model has the ability to return a probability distribution for each activity/movie by running the inference procedure multiple times. This enables us to quantify the prediction uncertainty of our model.

3.3.2 Transformer Encoder

Transformer (Vaswani et al., 2017) has overwhelmed the natural language processing world by its application of attention mechanism and its architecture. In our task, we will only utilize the encoder part of transformer which allows cross interaction of the information within the sequence level. Here we show in Figure 11 the architecture of our transformer model: here a single encoder block without residual connection is detailed and followed by a sum-up layer and output layer.

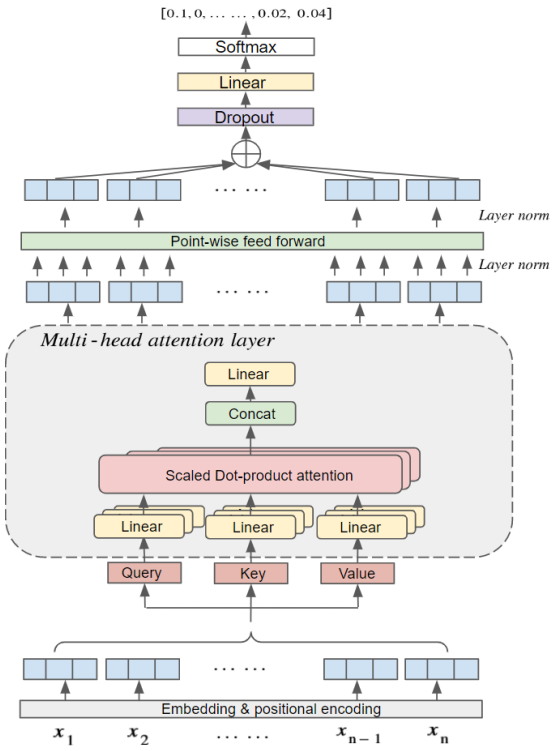

Figure 11: Transformer model structure

Positional Encoding When sequences of tokens are put into RNNs, its positional information is included. However, transformer encoder takes the input in parallel. To make positional information effective, transformer applies a positional encoding layer which has fixed values that only depend on their positions in the sequence. It is added to the embedding vector

thus their similarity in meaning and positions in the sequence account. As implemented in Vaswani et al. (2017), we will also apply trigonometric functions to encode positions for tokens in the sequence:

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d})$$
$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d})$$

where pos: position of token in the sequence, d: dimension of the embedding vector, i: the index of the vector. The corresponding encoded values for different (pos, i) pairs:

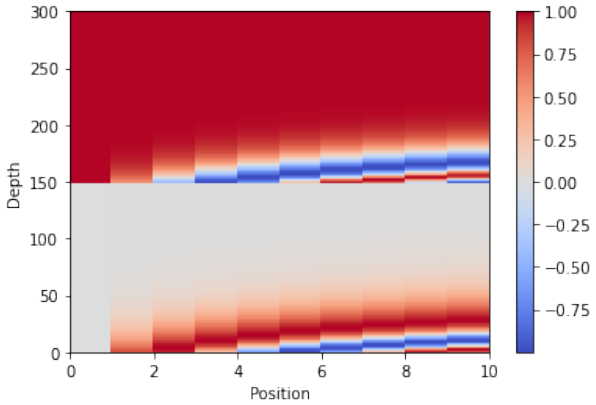

Figure 12: Positional encoding

Dot-Product Attention The scaled dot-product attention is defined as:

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}})\boldsymbol{V} \tag{2}$$

where $\boldsymbol{Q}$,$\boldsymbol{K}$,$\boldsymbol{V}$ represent queries, keys and values respectively. The Query-Key dot product computes the relevance between each pair of tokens. It is then used to compute a weighted sum of all the Values. And this weighted sum expresses the attention score for a (q,k,v) combination (matrix form means all combinations). The $\sqrt{d}$ is used to avoid exploding inner product when the embedding dimension is large.

Before taking the softmax of the Query-Key matrix, the mask is added into the matrix. The mask has the same shape as Q-K product and its entries are 1 where the padding covers and 0 where non-padding values are. This masking matrix is multiplied by negative infinity and added to the Q-K matrix, then cells with negative infinity are flushed out since softmax of negative infinity leads to zero.

Multi-Head Attention(MHA) In the transformer encoder, we will apply multi-head self-attention. In this setting, we divide the attention layer into m (divisible by embedding dimension) independently learned attention heads and feed queries, keys and values into these heads in parallel. And inside each head, independent attention scores are computed and scores from these heads are then concatenated and connected by a linear layer to produce the output.

- Inputs: Query $\boldsymbol{Q} \in \mathbb{R}^{n \times d_q}$, Key $\boldsymbol{K} \in \mathbb{R}^{n \times d_k}$, Value $\boldsymbol{V} \in \mathbb{R}^{n \times d_v}$. Since here we are using self-attention, we donote all inputs by $\boldsymbol{H} \in \mathbb{R}^{n \times d}$.
- Attention heads:

  $$h_i = f(\boldsymbol{W}_i^Q \boldsymbol{H}, \boldsymbol{W}_i^K \boldsymbol{H}, \boldsymbol{W}_i^V \boldsymbol{H})$$

  where f is the dot-produce attention (2).
- MHA output:

  $$\boldsymbol{Z} = \boldsymbol{W} \begin{pmatrix} h_1 \\ \vdots \\ h_m \end{pmatrix} \tag{3}$$

Point-Wise Feed-Forward Networks(FFN) The MHA block is able to incorporate cross information with adapative weights from previous embeddings, but the concatenated attention score is still a linear combination. To equip the model with non-linear property, a point-wise feed-forward network with one hidden layer (Hornik et al., 1989) is applied to all

output with identical weights:

$$F_i = ReLU(\boldsymbol{Z}_i\boldsymbol{W}^1 + \boldsymbol{b}^1)\boldsymbol{W}^2 + \boldsymbol{b}^2 \tag{4}$$

where $\boldsymbol{W}^1, \boldsymbol{W}^2, \boldsymbol{b}^1, \boldsymbol{b}^2$ are trainable FFN parameters.

Layer Normalisation Similar as batch normalisation, layer normalisation is used to along features which helps stabilise and boost training process (Ba et al., 2016). Normalisation layer is applied immediately after MHA and FFN layers. Different from batch normalisation, layer normalisation does not use statistics from other samples in the same batch.

$$LN(z) = \alpha \odot \frac{z - \mu}{\sqrt{\sigma^2 + \varepsilon}} + \beta \tag{5}$$

where $\odot$ is the Hadamard product, μ and σ^2 are mean and variance of z respectively. α and β are scaling factors and bias.

Residual Connections Instead of feeding the input sequentially into MHA and FFN layers, transformer encoder applies residual connections.

$$g(z) = z + Dropout(f(LN(z))) \tag{6}$$

where f can either be MHA (3) or FFN (4), dropout is defined in 3.3.1. With the residual connections, every time we feed normalised input z (except for the input from embedding-positional encoding layer) into the MHA or FFN, the output adds back the input z. The residual connections allow the input to directly flow into the output by skipping some layers (if these layers are not feasible). In many scenarios, we would need not only the processed output from non-linear mappings, but also "raw" input from low-layer level. For instances, in sequential recommendation, the next predicted token depends largely on information from last token. But this last token has communication with previous tokens in the MHA part, so adding back the last token's representation can help leverage information from lower level.

3.3.3 Output layer

Multilayer Perceptron(MLP) The vanilla output layer that bridges the output from LSTM/Transformer model to final prediction is multilayer perceptron. We can add hidden layers to approximate possible non-linear relationship or compute a final linear combination. Empirically, we found that a single linear layer helps prevent overfitting and leads to quicker training.

Softmax Activation In the final output layer, we will have K neurons that corresponds to padding (activity 0) and $K-1$ activities. And then we apply softmax function to transform these logits to probabilities. The i-th entry of the output vector the probability of i-th class.

The softmax activation function is shown as below:

$$softmax(\mathrm{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

where z is the output of the previous layer in the neural network.

Therefore, applying softmax gives interpretability of the result that the class with the higher logit value is more likely to be predicted. To make a prediction, we can use two methods:

- argmax: choose the one with the highest probability as the prediction.
- generate from distribution: each time, treat these probabilities as multinomial and choose one from the distribution.

3.4 Training and testing

Categorical cross-entropy Since our classification task consists of 131 classes, Categorical cross-entropy is used as the loss function. given each ground truth label y_i and prediction $\hat{y}_i$,

the Categorical cross-entropy loss is defined as:

$$L = -\sum_{k=1}^{K} y_{i_k} \log(\hat{y}_{i_k}), \quad K = 131$$

where y_{i_k} and $\hat{y}_{i_k}$ are the k-th scalar value of y_i and $\hat{y}_i$. A mask was added to the loss function in order to exclude loss generated from padding value zero.

Label smoothing To prevent models from overfitting, label smoothing is used as a regularisation technique during loss calculation. This is achieved through softening the one-hot encoded ground truth vector y_i by a value of α_{smooth}. The obtained new ground truth label is

$$y_{soft_i} = (1 - \alpha_{smooth})y_i + \frac{\alpha_{smooth}}{K}.$$

During training, the model will be less confident about its results and produce a smoother loss curve, which stabilise the training.

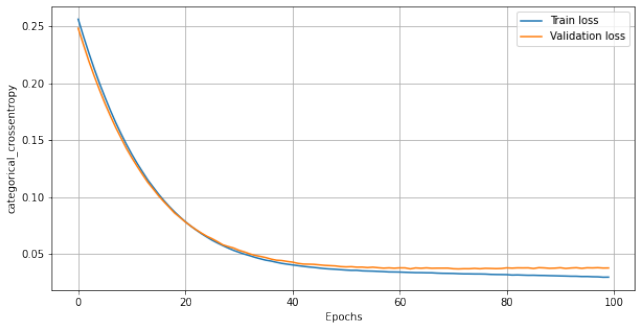

Figure 13: Bi-LSTM model trained on Koa data with Doc2Vec embedding.
Using both the regularisation in Figure 10 and Label smoothing of $\alpha_{smooth} = 0.1$.

Testing and Prediction For each user sequence in the test set S, we used the model obtained to generate N predictions $\hat{y}_{i_n}$, where each prediction is a K-dimensional softmax vector that sums to 1. For each scalar value $\hat{y}_{i_{n_k}}$ in each prediction $\hat{y}_{i_n}$, we calculate the mean for the k-th activity:

$$\bar{y}_{i_k} = \frac{\sum_{n=1}^{N} \hat{y}_{i_{n_k}}}{N}$$

and variance:

$$\mathrm{Var}(\hat{y}_{i_k}) = \frac{\sum_{n=1}^{N}(\hat{y}_{i_{n_k}} - \bar{y}_{i_k})^2}{N}$$

for each user. The activities/movies with the top k highest $\bar{y}_{i_k}$ will be recommended to the user. As $N \to \infty$, $\bar{y}_{i_k} \to \mu_{i_k}$ and $\mathrm{Var}(\hat{y}_{i_k}) \to \sigma^2_{i_{ik}}$, where μ_{i_k} and $\sigma^2_{i_k}$ are the true mean and variance of $\hat{y}_{i_k}$. In our experiment, we chose N = 500 to balance between accuracy and computational cost.

3.5 Evaluation Metrics

After model construction, we will use the test set split from the dataset to evaluate the prediction of the model. Since the Koa data is not balanced in target, in order to eliminate the difference in results of different test data, cross evaluation is applied to quantify the performance of the model.

Each dataset will be split into five equal-length mutually exclusive subsets, or called folds. For each dataset, we will train the model five times. For each training, four subsets are selected as the training set, and the remaining one is used as the test set. The overall performance of the model is calculated by the average statistic of five test sets.

3.5.1 Recall@k

After constructing the model, we will use the Monte Carlo method to estimate the expectation and variance of the user's reaction towards the activity. Activities with highest estimated mean reward will be the top-k recommended activities.

Hence, for each user, we will calculate the recall of the prediction:

$$I(y_i \in R_i) = \begin{cases} 0, y_i \notin R_i \\ 1, y_i \in R_i \end{cases}$$

where R is the top-k recommended activities of the user predicted by the model, y_i is the target activity of the user.

Then for all test users, we could have the overall recall:

$$\text{Recall@k}(S) = \frac{1}{|S|} \sum_{i \in S} I(y_i \in R_i)$$

where S is the set of test users.

In the project, recall@2 is regarded as the most important recall metric during evaluation. This is because Foundations app will recommend 2 activities in a row.

3.5.2 Soft Evaluation

Unlike direct evaluations which utilise recall or precision, we propose a soft evaluation metric called **AreaScore**. It can serve as an alternative evaluation standard for recommendation system where input data can be categorized. As in the two datasets we are using, Koa data has 11 focus areas where each focus area represents a category while MovieLens data has 18 genres, each genre represents the movie type. For each activity/movie, it can belong to at least one focus area/movie genre.

Intuitively, sometimes we might not be able to provide users with the item they like most, but with an item of the same category/genre. Shall we count that as a bad recommendation? Instead of using hard evaluation to assess the model performance, a softer metric may fill the gap when it is difficult to recommend the most accurate item for users. For examples, if one user truly favours movies that mutually belong to Action, Crime and War, then it is meaningful to recommend movies that also belong to the three genres, from a recommendation system's perspective.

The AreaScore is defined as:

$$\text{AreaScore}(S) = \sum_{i \in S} \frac{\#[C(\hat{y}_i) \cap C(y_i)]}{\max(\#(C(\hat{y}_i)), \#(C(y_i)))} / |S| \tag{7}$$

where $C(y) : Z^+ \rightarrow E$ is a mapping from class label to the set of events. E is the set of categories that y belongs to. Intuitively, we count the overlapping number of categories between prediction $\hat{y}_i$ and y_i, and divided by the maximum of the number of elements of the two sets. This makes sure that only two items belong to exactly the same categories then the AreaScore is 1. And we average these scores over A to give the final AreaScore on this data set. Hence, this soft evaluation approach serves as a mitigation between the exactly hard evaluation such as recall@1 and some loose metric such as recall@k (especially when k is large).

3.5.3 Coverage

Coverage represents the percentage of activities that the recommender system was able to recommend. For users in test set, we predict the most possible activity for the user to click. Then form a list R of all predicted unique activities:

$$\text{Coverage}(S) = \frac{1}{|S|} \sum_{i \in S} \frac{|R_i|}{K}$$

where R is the set of predicted activities for each user, and K is the number of classes of activities. The higher the value is, the more comprehensive the recommender system is.

3.5.4 Average Variance

Average variance evaluates how certain is the model in predicting the probabilities of activities that could be clicked by users. During prediction, we calculated the variance of each user in the test set for each activity, which was denoted as $\text{Var}(\hat{y}_{i_k})$ in Testing and Prediction of 3.4.

The average variance of the whole test set can be calculated as follow:

$$\mathrm{Var}(S) = \frac{\sum_{i \in S} \sum_{k \in K} \mathrm{Var}(\hat{y}_{i_k})}{|S|K}$$

where S denotes users in the test set and K is the number of all activities/movies in the vocabulary set. A relatively lower average variance indicates that a model is more confident about its predictions.

3.5.5 Cross-Evaluation

Figure 14: Cross-evaluation on Koa data set

Since Koa data has relatively small size and imbalanced distribution of activities, a single train/test can be quite volatile. We propose to use 5-fold cross-evaluation for Koa data. The corresponding ratio for train, validation and test sets is 8:1:1. This particular ratio was chosen because it produces the most stable training and returned the highest recall on the test set. For MovieLens data, it is reasonable to use random train/test split because it has much larger data size and less unbiased, we have experimented with random split and results are much more stable compared to Koa. We do this random split 5 times and average the performances to give the final statistics.

4 Experiment

4.1 Configuration & Results

In this section, we show our model configuration and training settings. We also present empirical results on the two datasets. We divide our systems into two groups:

Non-sequential recommendation

- **PopAct baseline**: A popularity-based recommender system with no personalisation. Five activities/movies with the highest frequency in the training set were selected. For each user in the testing set, one of the five activities was randomly assigned to this user. If we were to calculate recall@k where k>1, the activities/movies were randomly sampled without replacement.
- **Feedforward model**: A shallow fully connected network that could be used to discover non-linearity. We add one hidden layer (256 neurons) with ReLU activation after embedding layer.

Sequential recommendation

- **Bi-LSTM model**: We tune different latent dimension d for three various embedding methods. $d_{Act2Vec} = 64,\ d_{Doc2Vec} = 256,\ d_{SBERT} = 512.$
- **Transformer model**: The FFN dimension we choose according to the embedding used. $dff_{Act2Vec} = 64,\ dff_{Doc2Vec} = 128,\ dff_{SBERT} = 256$

Due to distinguishable data set size between Koa and MovieLens, we apply different settings.

Koa: Maximum sequence length 10. Adam optimizer (Kingma and Ba, 2014), drop out rate is 0.2, batch size of 32, training for 100 epochs. All evaluation metrics are based on 5-fold

cross-evaluation. Label smoothing with factor 0.1 is used, LSTM uses elastic regularizer for recurrent weights. Transformer model use one encoder block.

MovieLens: Maximum sequence length 10 with focus area concatenation. Adam optimizer, batch size of 64, dropout rate is 0.3, training for 50 epochs. All evaluation metrics are based on 5 random train/test splits. Only SBERT embedding utilizes label smoothing due to larger latent dimension. Transformer stacks two encoders.

Dataset	Metric	PopAct	Feedforward model				Bi-LSTM model				Transformer model			
		baseline	One-hot	Act2Vec	Doc2Vec	SBERT	One-hot	Act2Vec	Doc2Vec	SBERT	One-hot	Act2Vec	Doc2Vec	SBERT
Koa	recall@1	2.90%	3.68%	3.75%	9.06%	2.18%	4.37%	5.31%	**9.06%**	5%	6.56%	7.71%	6.67%	5.63%
	recall@2	7.18%	6.31%	7.81%	13.12%	7.18%	6.87%	10.31%	**13.13%**	10%	8.37%	11.45%	11.98%	8.75%
	recall@5	10.84%	12.18%	16.25%	20.93%	14.37%	16.56%	19.37%	**22.26%**	20.93%	15.93%	17.50%	18.44%	15.63%
	AreaScore	-	29.45%	32.85%	**39.23%**	32.32%	33.11%	31.45%	36.12%	29.08%	31.82%	32.15%	31.45%	33.20%
	coverage@1	-	1.3%	6%	1.2%	4.92%	6.3%	1.53%	7.9%	10.77%	4.5%	1.85%	5.44%	**13.08%**
	avg. variance	-	**2.88e-8**	2.58e-4	2.88e-5	3.87e-4	1.24e-5	1.25e-5	3.31e-5	7.54e-5	2.5e-3	2.28e-5	4.36e-3	4.61e-3
Movie-Lens	recall@1	2.12%	3.88%	5.42%	3.15%	4.99%	5.41%	7.19%	9.02%	9.93%	6.01%	6.09%	**11.57%**	9.26%
	recall@2	3.53%	5.25%	8.85%	5.70%	8.45%	9.25%	11.61%	13.31%	14.74%	10.41%	10.44%	**16.86%**	13.65%
	recall@5	6.33%	10.18%	16.3%	11.29%	15.95%	17.59%	21.12%	22.47%	25.01%	20.19%	21.28%	**26.41%**	23.01%
	AreaScore	-	25.43%	27.99%	24.96%	27.51%	28.11%	29.95%	30.31%	31.79%	28.75%	30.30%	**32.86%**	29.54%
	coverage@1	-	32%	38.46%	80%	**91.15%**	73.8%	84.07%	83.84%	84.84%	56.48%	48.46%	70.76%	76.92%
	avg. variance	-	**1.11e-6**	2.72e-5	2.26e-4	1.13e-4	2.34e-5	4.71e-5	5.45e-5	5.37e-5	4.67e-5	4.09e-5	4.29e-5	4.15e-5

Table 7: Recommendation Performances on 6 different evaluation metrics. The best statistic in each metric is boldfaced

4.2 Model Improvement

Initially we try the simple feedforward neural network, but it can only learn the latent non-linear relationship between inputs and outputs, important sequential information is missed. Intuitively, in order to represent an user by past activities, the order matters. The last activity usually carries more information than others, hence we consider using sequential models including Bi-LSTM and transformer. To represent improvement by designing these models, we use a baseline called PopAct, which is a popular baseline to be compared in recommendation system. On Koa dataset, the figures of Bi-LSTM model with Doc2Vec at least doubled that of baseline in terms of three recalls. We can also find feedforward model with Doc2Vec obtained similar results as Bi-LSTM on all metrics except coverage. However, feedforward failed on diversifying its predictions where the coverage@1 only hits 1.2%, which

means it almost predicts identical activity with the highest probability for every user. On the contrary, Bi-LSTM model achieved slightly better performances with 8% coverage, which is a fair number on this dataset. Transformer does not behave well on this dataset, except that it obtained the most diversified predictions and sound AreaScore when using SBERT embedding.

On MovieLens dataset, PopAct baseline performed worse than on Koa dataset, because MovieLens is more balanced. Feedforward model achieved similar recalls with Act2Vec and SBERT embedding, but failed on Doc2Vec. Bi-LSTM encircled feedforward in almost all statistics. Furthermore, transformer performed even better than Bi-LSTM. Transformer with Doc2Vec leads on three recalls and AreaScore, by quadrupling PopAct and doubling feedforward. It also is 2% higher on these metrics than best performed Bi-LSTM model. In terms of coverage, all models have much better figures than on Koa dataset. Transformer with Doc2Vec can achieve the highest accuracy while maintaining a good coverage. It can also be seen that model variance becomes smaller, while transformer is less variant than Bi-LSTM and feedforward when using three adaptive embeddings.

4.3 Embedding Analysis

Table 7 presents the model performance of all methods on the two datasets by four different embedding methods. By not extracting any information from the movie, one-hot encoding is used as a baseline for embedding methods. Besides the intuitive baseline, Act2Vec, Doc2Vec and SBERT embedding are also applied. From Table 7, it can be found that in almost all the situations, the proposed embedding methods could improve the performance of the model.

In Koa data, it could be found that Doc2Vec has the best performance in recall for all the proposed models. Recall@5 for Doc2Vec is at least 6% better than the baseline. On the other hand, high accuracy is not the only criteria for recommender system. Doc2Vec embedding could have better performance in recall but in coverage level, it doesn't perform as well as other methods. In feedforward model, which is non-sequential, Act2Vec has the highest

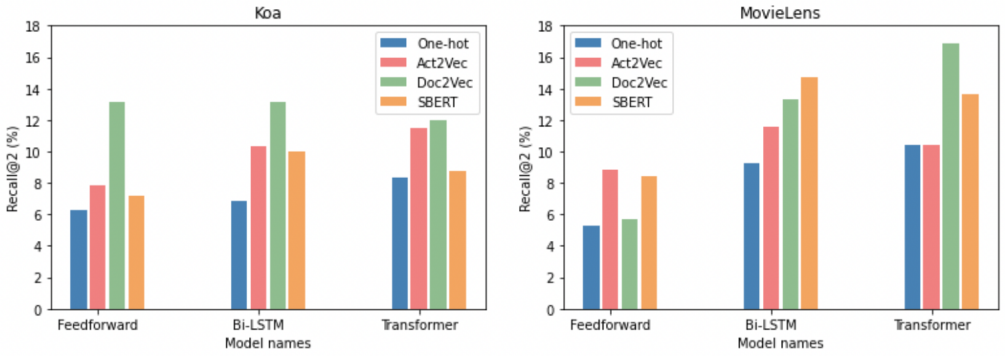

Figure 15: Embedding Evaluation

coverage score while in sequential models, SBERT performs better in the scale of coverage.

Conversely, in MovieLens data, Doc2Vec's performance varies among all the models as presented in Figure 15. It has the worst performance in feedforward model, where just outperforms the baseline by just 1%. However, it performs the best in the transformer model. Among all the embedding, SBERT performs the best by taking recall level and coverage level metric into account. It is worth noting that in feedforward model, the coverage of SBERT embedding is more than 90%, which indicates the embedding helps the model to perform more comprehensive rather than only predict the dominant target.

Overall, the simple embedding could perform as well as the complex embedding in simple models. As the model structure becomes more complex, more information needs to be provided by the embedding. In the experiment, we found that Doc2Vec and SBERT embedding have better performance in sequential models. In conclusion, the proposed embedding methods are all providing more information than the baseline one-hot embedding. And for different model, the same embedding methods could have different performance. Even for a simple model, if the embedding is chosen appropriately, it could have comparable performance to complex models.

4.4 Uncertainty

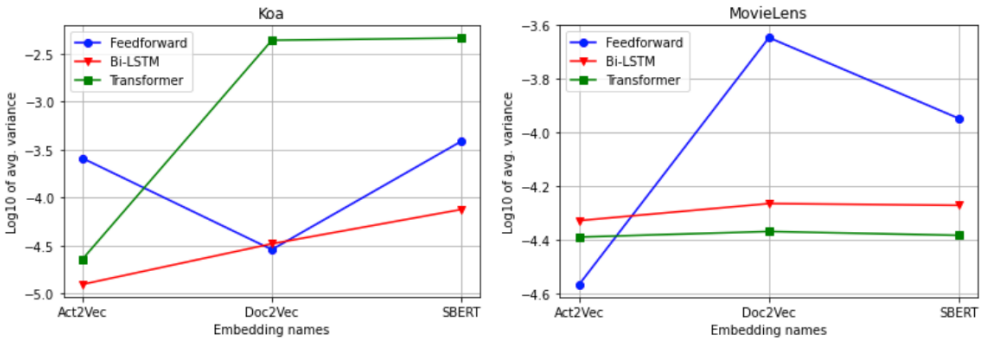

Figure 16: Average variances on logarithmic scale

Figure 16 shows the average variance in section 3.5.4 produced by our models on a logarithmic scale. We noticed that simpler model, such as the feedforward model, doesn't necessarily produce more certain predictions than models with more complex architectures. On the smaller Koa dataset, The Bi-LSTM model produced more stable results compared to other models, while model uncertainty of the transformer drastically increased with increasing embedding dimension. The feedforward model has the lowest level of uncertainty when Doc2Vec embedding is applied.

On the larger MovieLens dataset, both the Bi-LSTM model and transformer were able to produce stable predictions regardless of embedding types, with transformer having the lowest level of uncertainty when using Doc2Vec or SBERT embeddings. Although the feedforward model had the lowest uncertainty level on the Koa dataset, it was not able to remain stable on MovieLens data, and its uncertainty fluctuated with changing embedding types.

To conclude, the Bi-LSTM model has been the most stable one regardless of the size of data and embedding types. Model uncertainty of feedforward model fluctuated with different data and embedding types. Transformer's stability increased on a relatively larger dataset.

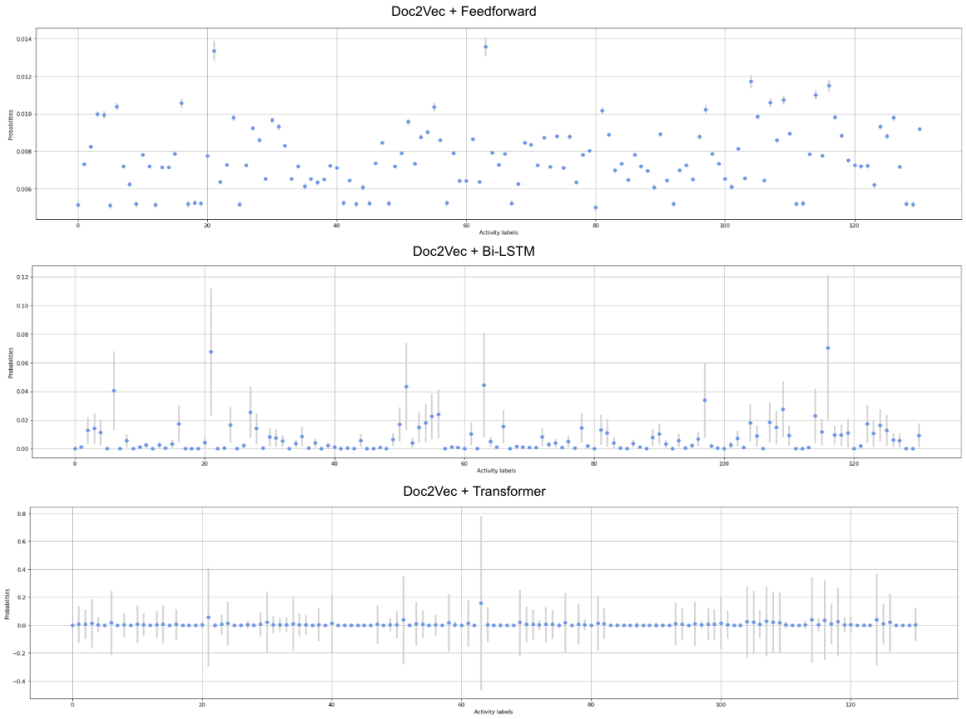

Figure 17: Uncertainty plots of user 2 in Koa's testing set (label 0 represents padding value 0).

Figure 17 illustrates an example of uncertainty visualisation for a test user. Blue dots correspond to the mean probabilities of each of recommendation, and grey bars define the standard deviations of the predicted probabilities of each activity. In Figure 16, the feedforward model has the lowest uncertainty with Doc2Vec embedding on the Koa dataset, followed by Bi-LSTM model and transformer. For this particular user in Koa's testing set, we compare the uncertainty of these three different types of model using Doc2Vec embedding.

Looking at the plots, we noticed that activities predicted from the feedforward model have the shortest standard deviation bars, meaning the predicted probabilities are more certain and stable for each prediction. As the model uncertainty increases, the standard deviation for each predicted activity has increased. There are also more overlaps between the standard deviation bars as the level of uncertainty increases. Notably, the activities with the highest predicted probabilities have the highest standard deviation.

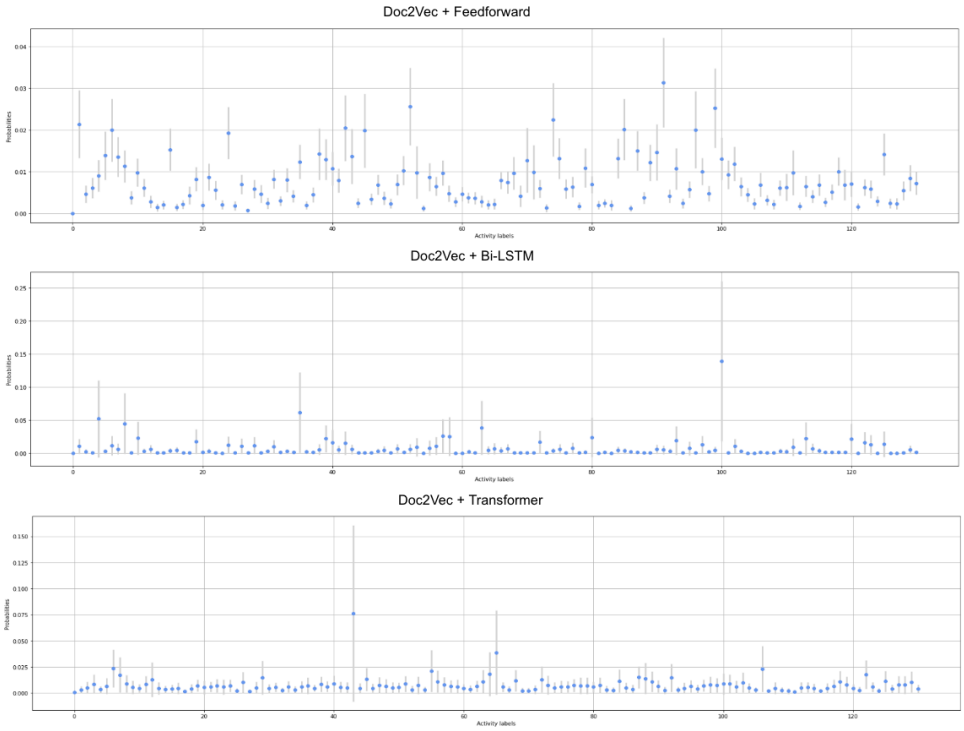

Figure 18: Uncertainty plots of user 2 in MovieLens' testing set.

Figure 18 represents the prediction uncertainty of a particular user from the MovieLens dataset. For this particular user, both the Bi-LSTM model and the transformer have returned a relatively stable estimate, with uncertainty mainly concentrated on few activities. For the feedforward model, there are many visible overlaps between activities, indicating a relatively high level of uncertainty in the prediction.

5 Discussion

In this work, we proposed self-attention Bi-LSTM model and multi-head attention transformer model for the next activity recommendation. Unlike the baseline model, all proposed models only take 10 latest activities of a user for prediction, which guarantees the personalisation of the system. We have reach the state of art result. The performance of the proposed methods has been discussed in Section 4 that personalized model is evaluated better than the non-personalised popular activity baseline. For Koa data, the best model and embedding pair outperforms baseline model by 6% on recall@2. In MovieLens data, the best model exceeds the baseline for 13%. The coverage on MovieLens dataset is larger than on Koa dataset. This might be caused by dataset imbalance. The imbalance, or the dominant activity will lead to bias in the model.

Moreover, compared with collaborative RNN, our models also have outstanding performance. Fu's team apply a novel collaborative filtering based RNN model (Fu et al., 2018) on MovieLens-1M data and their model could obtain 28.16% on recall@10. Our best model could reach to 35.73% on a larger dataset. The embedding method we chose also help to improve the model performance compared to other similar models applied on MovieLens data which used Item2Vec embedding (Zhao et al., 2020). The Item2Vec embedding could only help the model to reach 22.63% recall@20.

Model:	**Transformer, Doc2Vec on filtered MovieLens**	**Collaborative filtering based RNN on 1M Movielens**	**Bi-LSTM, Item2Vec on 10M Movielens**
Recall@10	35.73%	28.16%	-
Recall@20	49.21%	-	22.63%

Table 8: Model comparison with existing research on MovieLens data

On the other hand, model with the highest recall value does not necessary means better model. Taking feedforward model with Doc2Vec embedding in Koa data as an example, it has a relatively high recall value but a smallest coverage rate among all models. It indicates

that the model only gives out 2 distinct activities among 130. Considering at this level, the model is biased by the dominant activity in the dataset.

As we expected, the Transformer model will have the best performance, which is verified on MovieLens dataset. While in Koa data, the result does not appear to meet our expectation. As mentioned in Section 3.1, Koa data has less than 700 records, which is easy for complex model to overfit. On the other hand, there are some dominant activities in the dataset, which means even the model have lower coverage, it could have a good performance on recall.

There are several notable parts in the proposed methodology. We not only consider the activity itself in the dataset, but also taking the category of the activity, focus area, into account. For the embedding chosen part, we proposed a self-defined loss function. In order to give out the distribution of the award, we applied dropout in the model to mimic the effect of Bayesian neural network. In the evaluation process, apart from the traditional hard evaluation method, we proposed the soft evaluation metric which is more suitable for the recommender system logic.

5.1 Limitation

Critically, findings from both the MovieLens and Koa datasets indicate the possibility for a more effective method of activity recommendation. The possible models for improvement will be discussed in Section 6. In the data level, controlled by the number of the activities in Koa dataset, the models are not applied on more sparse data, which might could give us more insights to the model. In both datasets, there is no information about the user rather than user ID. More information on user features could give model more information to learn the pattern of the user activities. The whole experiment is performed in an offline environment, which is different with the real-world online recommender system. Ideally, an A/B test could be implemented to assess the model with real-time user data and compare it with the recommender system currently used. However, the online implementation is out of the scope of the project.

6 Conclusions and future work

6.1 Koa Foundations App

In this work, we demonstrated that sequential models outperform non-sequential models in the task of personalised activity recommendation. In Koa dataset, the Doc2vec embedding has shown to have the best performance in terms of recall and coverage. In addition, we implemented dropout inference to achieve the effect of a Bayesian model, and concluded that Bi-LSTM model has the most stable level of confidence with its predictions.

The next possible step is to deploy our models into Koa's existing multi-armed bandits framework, and to test the overall impact it has on the general framework with a larger scale of data. As mentioned before, the Gaussian process model is used to provide a sampling pool for Thompson sampling. However, the Gaussian process is computationally expensive with a complexity of $O(n^3)$ comparing to $O(n)$ for dropout approximation. Hence, it is not suitable for larger scales of data. By replacing the Gaussian process with a Bayesian neural network, an estimation of the distributions can be obtained for each user and each activity at a lower computational cost. Figure 19 demonstrate the general framework of Koa's recommender system.

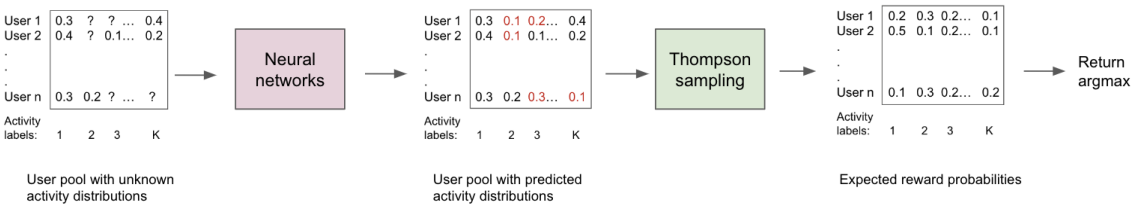

Figure 19: General framework of Koa's personalised activities recommender system.

6.2 Other Sequential Models

In our experiments, we have demonstrated the effectiveness of RNN and transformer models in providing sequential recommendations for mental health interventions. Despite the architectures proposed in our methodology, several other architectures within the transformer class will be worth exploring.

The BERT models have achieved outstanding performance in NLP tasks, with many successful attempts made to extend its usage to recommender systems. The training of BERT models usually requires a even larger corpus. Once enough data is generated, the BERT model can be potentially implemented on the Koa dataset.

The Generative Pre-trained Transformer 2 (GPT-2) (Radford et al., 2018) model was also developed for generating the next item in the sequence. In contrast to the BERT model, GPT-2 is an unsupervised transformer model that only consists of the decoder part, and is unidirectional in the attention part. It would be intriguing to experiment with these two architectures side by side in the future.

Bibliography

Atassi, A., Azami, I. E. and Sadiq, A. (2018), New gru from convolutional neural network and gated recurrent unit, *in* 'Proceedings of the First International Conference on Data Science, E-learning and Information Systems', pp. 1–2.

Attridge, M. (2019), 'A global perspective on promoting workplace mental health and the role of employee assistance programs'.

Ba, J. L., Kiros, J. R. and Hinton, G. E. (2016), 'Layer normalization', *arXiv preprint arXiv:1607.06450* .

Bowman, S. R., Angeli, G., Potts, C. and Manning, C. D. (2015), 'A large annotated corpus for learning natural language inference', *arXiv preprint arXiv:1508.05326* .

Chandrashekar, P. (2018), 'Do mental health mobile apps work: evidence and recommendations for designing high-efficacy mental health mobile apps', *Mhealth* **4**.

Chen, Q., Zhao, H., Li, W., Huang, P. and Ou, W. (2019), Behavior sequence transformer for e-commerce recommendation in alibaba, *in* 'Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data', pp. 1–4.

Damianou, A. and Lawrence, N. D. (2013), Deep gaussian processes, *in* 'Artificial intelligence and statistics', PMLR, pp. 207–215.

Feinson, M. C. and Popper, M. (1995), 'Does affordability affect mental health utilization? a united states-israel comparison of older adults', *Social science & medicine* **40**(5), 669–678.

Fort, S., Hu, H. and Lakshminarayanan, B. (2019), 'Deep ensembles: A loss landscape perspective', *arXiv preprint arXiv:1912.02757* .

Fu, M., Qu, H., Yi, Z., Lu, L. and Liu, Y. (2018), 'A novel deep learning-based collaborative filtering model for recommendation system', *IEEE transactions on cybernetics* **49**(3), 1084–1096.

Fuller-Tyszkiewicz, M., Richardson, B., Klein, B., Skouteris, H., Christensen, H., Austin, D., Castle, D., Mihalopoulos, C., O'Donnell, R., Arulkadacham, L. et al. (2018), 'A mobile app–based intervention for depression: End-user and expert usability testing study', *JMIR mental health* **5**(3), e9445.

Gal, Y. and Ghahramani, Z. (2016), Dropout as a bayesian approximation: Representing model uncertainty in deep learning, *in* 'international conference on machine learning', PMLR, pp. 1050–1059.

Gers, F. A., Schmidhuber, J. and Cummins, F. (2000), 'Learning to forget: Continual prediction with lstm', *Neural computation* **12**(10), 2451–2471.

Goldberg, S. B., Lam, S. U., Simonsson, O., Torous, J. and Sun, S. (2022), 'Mobile phone-based interventions for mental health: A systematic meta-review of 14 meta-analyses of randomized controlled trials', *PLOS Digital Health* **1**(1), e0000002.

Guo, C., Pleiss, G., Sun, Y. and Weinberger, K. Q. (2017), On calibration of modern neural networks, *in* 'International Conference on Machine Learning', PMLR, pp. 1321–1330.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. R. (2012), 'Improving neural networks by preventing co-adaptation of feature detectors', *arXiv preprint arXiv:1207.0580* .

Hochreiter, S. (1998), 'The vanishing gradient problem during learning recurrent neural nets and problem solutions', *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **6**(02), 107–116.

Hochreiter, S. and Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Hornik, K., Stinchcombe, M. and White, H. (1989), 'Multilayer feedforward networks are universal approximators', *Neural networks* **2**(5), 359–366.

Kendall, A. and Gal, Y. (2017), 'What uncertainties do we need in bayesian deep learning for computer vision?', *Advances in neural information processing systems* **30**.

Kingma, D. P. and Ba, J. (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

Kuyken, W., Warren, F. C., Taylor, R. S., Whalley, B., Crane, C., Bondolfi, G., Hayes, R., Huijbers, M., Ma, H., Schweizer, S. et al. (2016), 'Efficacy of mindfulness-based cognitive therapy in prevention of depressive relapse: an individual patient data meta-analysis from randomized trials', *JAMA psychiatry* **73**(6), 565–574.

Lakshminarayanan, B., Pritzel, A. and Blundell, C. (2017), 'Simple and scalable predictive uncertainty estimation using deep ensembles', *Advances in neural information processing systems* **30**.

Lau, J. H. and Baldwin, T. (2016), 'An empirical evaluation of doc2vec with practical insights into document embedding generation', *arXiv preprint arXiv:1607.05368* .

Le, Q. and Mikolov, T. (2014), Distributed representations of sentences and documents, *in* 'International conference on machine learning', PMLR, pp. 1188–1196.

LeCun, Y., Bengio, Y. and Hinton, G. (2015), 'Deep learning', *nature* **521**(7553), 436–444.

Liu, D., Farajalla, G. P. and Boulenger, A. (2021), Transformer-based banking products recommender system, *in* 'Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval', pp. 2641–2642.

Lu, J., Wu, D., Mao, M., Wang, W. and Zhang, G. (2015), 'Recommender system application developments: a survey', *Decision Support Systems* **74**, 12–32.

Luong, M.-T., Pham, H. and Manning, C. D. (2015), 'Effective approaches to attention-based neural machine translation', *arXiv preprint arXiv:1508.04025* .

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P. and Wilson, A. G. (2019), 'A simple baseline for bayesian uncertainty in deep learning', *Advances in Neural Information Processing Systems* **32**.

Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013), 'Efficient estimation of word representations in vector space', *arXiv preprint arXiv:1301.3781* .

Molloy, A. and Anderson, P. L. (2021), 'Engagement with mobile health interventions for depression: A systematic review', *Internet Interventions* **26**, 100454.

Movielens 20m dataset (2021).
URL: *https://grouplens.org/datasets/movielens/20m/*

Naeini, M. P., Cooper, G. and Hauskrecht, M. (2015), Obtaining well calibrated probabilities using bayesian binning, *in* 'Twenty-Ninth AAAI Conference on Artificial Intelligence'.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B. and Snoek, J. (2019), 'Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift', *Advances in neural information processing systems* **32**.

Park, Y.-T. (2016), 'Emerging new era of mobile health technologies', *Healthcare informatics research* **22**(4), 253–254.

Pazzani, M. J. and Billsus, D. (2007), Content-based recommendation systems, *in* 'The adaptive web', Springer, pp. 325–341.

Platt, J. et al. (1999), 'Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods', *Advances in large margin classifiers* **10**(3), 61–74.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I. et al. (2018), 'Improving language understanding by generative pre-training'.

Reimers, N. and Gurevych, I. (2019), 'Sentence-bert: Sentence embeddings using siamese bert-networks', *arXiv preprint arXiv:1908.10084* .

Ricci, F., Rokach, L. and Shapira, B. (2011), Introduction to recommender systems handbook, *in* 'Recommender systems handbook', Springer, pp. 1–35.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986), 'Learning representations by back-propagating errors', *nature* **323**(6088), 533–536.

Schafer, J. B., Frankowski, D., Herlocker, J. and Sen, S. (2007), Collaborative filtering recommender systems, *in* 'The adaptive web', Springer, pp. 291–324.

Schuster, M. and Paliwal, K. K. (1997), 'Bidirectional recurrent neural networks', *IEEE transactions on Signal Processing* **45**(11), 2673–2681.

Shen, X., Yi, B., Zhang, Z., Shu, J. and Liu, H. (2016), Automatic recommendation technology for learning resources with convolutional neural network, *in* '2016 international symposium on educational technology (ISET)', IEEE, pp. 30–34.

Skovhoj, F. Z. (2022), 'Using collaborative filtering in e-commerce: Advantages amp; disadvantages'. **URL:** *https://blog.clerk.io/collaborative-filtering*

Thompson, W. R. (1933), 'On the likelihood that one unknown probability exceeds another in view of the evidence of two samples', *Biometrika* **25**(3-4), 285–294.

Turkin, A. (2017), 'Tikhonov regularization for long short-term memory networks', *arXiv preprint arXiv:1708.02979* .

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. (2017), 'Attention is all you need', *Advances in neural information processing systems* **30**.

Wang, H., Wang, N. and Yeung, D.-Y. (2015), Collaborative deep learning for recommender systems, *in* 'Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining', pp. 1235–1244.

Williams, A., Nangia, N. and Bowman, S. R. (2017), 'A broad-coverage challenge corpus for sentence understanding through inference', *arXiv preprint arXiv:1704.05426* .

Williams, C. K. and Rasmussen, C. E. (2006), *Gaussian processes for machine learning*, Vol. 2, MIT press Cambridge, MA.

Wu, S., Ren, W., Yu, C., Chen, G., Zhang, D. and Zhu, J. (2016), Personal recommendation using deep recurrent neural networks in netease, *in* '2016 IEEE 32nd international conference on data engineering (ICDE)', IEEE, pp. 1218–1229.

Yadav, N. and Singh, A. K. (2020), Bi-directional encoder representation of transformer model for sequential music recommender system, *in* 'Forum for Information Retrieval Evaluation', pp. 49–53.

Zaremba, W., Sutskever, I. and Vinyals, O. (2014), 'Recurrent neural network regularization', *arXiv preprint arXiv:1409.2329* .

Zhang, S., Yao, L., Sun, A. and Tay, Y. (2019), 'Deep learning based recommender system: A survey and new perspectives', *ACM Computing Surveys (CSUR)* **52**(1), 1–38.

Zhao, C., You, J., Wen, X. and Li, X. (2020), 'Deep bi-lstm networks for sequential recommendation', *Entropy* **22**(8), 870.