Abstract

BERT4REC-BASED SERVICE RECOMMENDATION ENGINE FOR AZURE MARKETING

Shariq Ahmad, CX Data, shariqahmad@microsoft.com
Kirk Li, CX Data, kirk.li@microsoft.com

Keywords. BERT4Rec, attention, service recommendation, Azure portal.

Recommending appropriate services to the customer is a challenging task as there are a wide variety of Azure services. The Marketing team leverages the Azure Service Recommendation model for the email campaign drive and Azure portal. Traditional recommendation systems utilize a Matrix Factorization (MF) to uncover latent dimensions representing users and services properties. The challenge with MF is that it captures the static preference of the users, but it fails to analyze the dynamic preference of the customers. Customers using Azure services tend to prioritize the last few recent actions. It is quite crucial to recommend services based on the user dynamic preference from their recent historical behavior. To handle sequential dynamics, we propose a novel BERT4Rec-based model for Azure Service Recommendation that recommends services based on user history and significantly outperforms other recommendation system approaches. The proposed BERT4Rec-based model also provides the reason for recommending a service by taking advantage of the embedding layer and self-attention mechanism to understand the correlation between the services in the sequence. To avoid the cold start problem for new users, it leverages some of the users features.

1. Introduction

The task of recommending appropriate Azure services is a complex task due to the extensive range of services. The Azure Service Recommendation model plays a vital role in suggesting the top 5 out of 100 distinct services for every million users. This recommendation system is leveraged by the Marketing team to drive email campaigns and is integrated into the Azure portal. Customers using Azure services tend to prioritize the last few historical actions. It's quite crucial for the Azure Service Recommendation model to recommend services based on the user's dynamic preferences from their historical behavior.

Traditional recommendation systems utilizes Matrix Factorization (MF) to uncover latent dimensions representing users and services properties. The challenge with the MF approach is that it captures the static preference of the user, but it fails to analyze the dynamic preference of the customers. Moreover, the MF encounters difficulties in addressing the cold start problem when dealing with new users. This issue arises because the MF cannot incorporate user content information necessary for recommending services to users who have no prior interaction history.

Sequential dynamics tend to capture the context of users based on recent actions performed by them. Several methods have been proposed to address these sequential dynamics and provide recommendations based on users' historical interactions. The ultimate goal of sequential recommendation is to integrate users' static behavior with contextual information derived from their recent actions. Markov Chains (MC) assume that the next action is conditioned only on the previous action, but it fails to capture the sequential dynamics of a more complex scenario. While Recurrent Neural Networks (RNNs) like GRU and LSTM have shown more promising results compared to MC, they still lag behind Transformer-based models [1]. Also, RNNs are slow for inference due to their inability to perform parallel computations. In contrast, Transformer models like BERT and GPT have demonstrated significant advancements in various sequential tasks by utilizing attention mechanisms to learn sequential dependencies. Additionally, the Transformer models run much faster as they support parallel acceleration.

Therefore, we propose a novel BERT4Rec-based model for Azure Service Recommendation that utilizes Transformer Encoder [1] for recommending a service. In Section 5.4, we analyzed various components of the Transformer, including Multi-head Attention layers and Feed Forward layers. Our findings revealed that increasing the number of attention heads and the dimension of Attention layers significantly enhances the model's performance.

The main contribution of this paper is as follows:

1. Proposal of a novel BERT4Rec-based model for Azure Service Recommendation: Our model incorporates an attention-based mechanism to effectively handle the sequential dynamics of user behavior, outperforming traditional approaches.
2. Provision of service recommendation reasoning: The proposed BERT4Rec model not only recommends services but also provides the underlying reasons for making those recommendations.
3. Handling of new users and mitigation of the cold-start problem by leveraging content features.
4. The model is computationally fast.

2. Related work

2.1. General recommendation

Early work on recommendation systems primarily relied on Collaborative Filtering [2] to gain insights into users' preferences. The most popular methods are Nearest Neighbor and Matrix Factorization (MF).

The Nearest Neighbor approach involved calculating user similarity based on the user-service matrix. It identified the top-k users who were most similar to a particular user and recommended services that were being used by similar users.

Matrix Factorization (MF) computed latent space vectors for both users and services, extracted from the user-service matrix. It estimated a user's preference for a service by taking the dot product of the user's latent space vector with the service's latent space vector. MF performed better than Nearest Neighbor by capturing the complex interactions between users and services. However, both approaches had a limitation: they failed to account for the dynamics of user preferences changing over a period of time.

2.2. Sequential recommendation

In the early stages of research on sequential recommendation, Markov Chains (MC) were commonly employed to capture the sequential patterns within users' historical interactions. Later, the power of MC was combined with MF to model sequential and general interests of users by Factorizing Personalized Markov Chains (FPMC) [3]. Recurrent neural networks (RNNs) have performed reasonably well in modeling sequences, but they may not work well for sequential recommendation since not all adjacent services have dependency relationships.

In recent advancements, Transformer-based models [1], such as BERT [4] and GPT [5], have emerged as state-of-the-art solutions in various sequential tasks, including machine translation. These models leverage the power of Multi-head Attention identifying the relevant components of the input that the model should focus on.

3. Data

The Azure Service Recommendation model relies on historical service usage data for each user ID, covering a period of 120 days. The daily usage data, ingested from MCDS Delta Lake consists of user ID, service, and usage. Additionally, the model incorporates Firmographic features, such as segment name, sub-segment, area, and industry. These Firmographic features play a crucial role in addressing the cold start problem, particularly for new users. The data pipeline is shown in Figure 1.

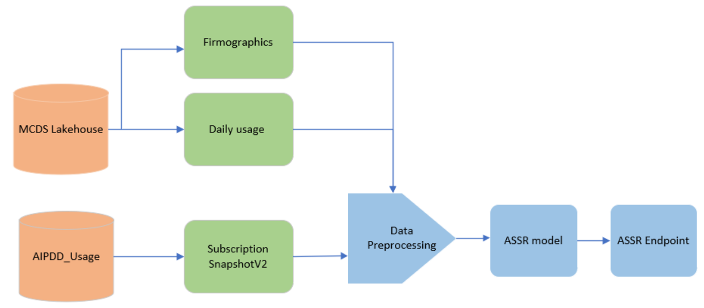
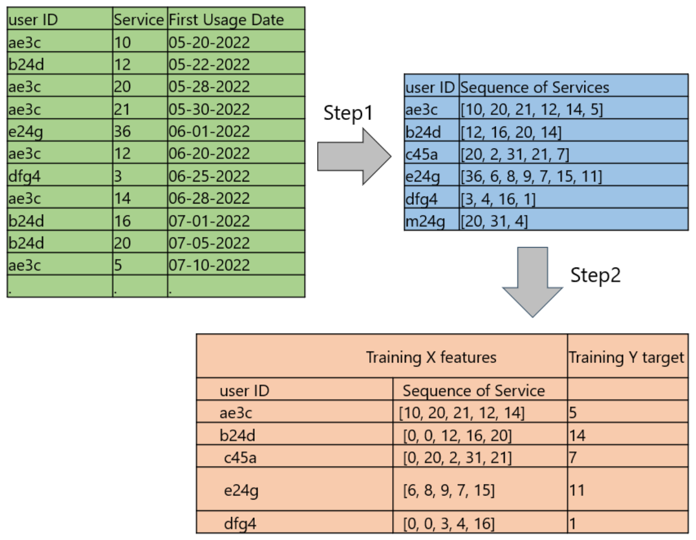


Figure 1: Data pipeline.

1. Filter out very low usage data: For training the model, we filter out user IDs having usage for all the services below a threshold value of 10. This will remove outliers from the data.
2. Convert the usage of users x service into a rating: The distribution of each service is plotted by utilizing usage data from all users who have utilized the service and provided ratings ranging from 1 to 5 by dividing the distribution into five equally sized bins. These ratings will be utilized by the loss function for training the BERT4Rec.
3. Label encoding of the categorical features: The user ID, service, and Firmographics data consisting of Segment, Sub-segment, Area and Industry are label encoded before feeding it into the model.
4. Converting the user history of services data into a sequence of services: For each service, we identify the first usage date when it was added by a specific user ID. This first usage date serves as a reference point for ordering the services used by that user ID. Using this approach, we convert the user's service history into a sequence where the services are arranged in ascending order based on their respective first usage dates. This conversion process is illustrated in Step 1 of Figure 2.
5. Preparation of features and target: All services, except the last one in the sequence, are used as the training feature set (train X). The last service in the sequence is assigned as the training target (train Y). In cases where the number of services used exceeds the specified sequence length, we consider only the last n sequences, where n represents the total sequence length. On the other hand, if the number of services is less than the sequence length, we pad the sequence with zeros on the left side. This padding ensures that the sequence length matches the desired length and maintains consistency during the training process as shown in Step 2 of Figure 2.



user ID	Service	First Usage Date
ae3c	10	05-20-2022
b24d	12	05-22-2022
ae3c	20	05-28-2022
ae3c	21	05-30-2022
e24g	36	06-01-2022
ae3c	12	06-20-2022
dfg4	3	06-25-2022
ae3c	14	06-28-2022
b24d	16	07-01-2022
b24d	20	07-05-2022
ae3c	5	07-10-2022
.	.	.

user ID	Sequence of Services
ae3c	[10, 20, 21, 12, 14, 5]
b24d	[12, 16, 20, 14]
c45a	[20, 2, 31, 21, 7]
e24g	[36, 6, 8, 9, 7, 15, 11]
dfg4	[3, 4, 16, 1]
m24g	[20, 31, 4]

Training X features		Training Y target
user ID	Sequence of Service	
ae3c	[10, 20, 21, 12, 14]	5
b24d	[0, 0, 12, 16, 20]	14
c45a	[0, 20, 2, 31, 21]	7
e24g	[6, 8, 9, 7, 15]	11
dfg4	[0, 0, 3, 4, 16]	1

Figure 2: (1) Conversion of user-services data into a sequence of services per user ID according to first usage date. (2) Preparation of features and target.

4. Methods

This section focuses on examining the baseline models and the proposed model for the recommendation system. Initially, we will employ baseline models such as Nearest Neighbour Collaborative Filtering and Neural Collaborative Filtering to gain insights into their limitations. Subsequently, we will delve into the proposed BERT4Rec-based model, exploring its architecture and the loss function employed for training.

4.1. Baseline models

k-Nearest Neighbor Collaborative Filtering: The k- Nearest Neighbor (kNN) collaborative filtering method operates by taking user ID, service, and corresponding usage as input. It further converts the usage data into a rating scale ranging from 1 to 5. During the training process, the algorithm focuses on deeply engaged users by filtering out a subset of users with a high level of service usage. The kNN algorithm is very sensitive to outliers, so it is essential to remove users having low usage from the training data as outliers adversely affect the recommendations. To facilitate the recommendation process, the user, service, and rating information is converted into a user-service matrix. In this matrix, each column represents a unique service, each row represents an individual user, and the corresponding matrix value indicates the rating given by the user for the service.

The kNN collaborative filtering algorithm finds the distance between the user ui and all other users in the training set. It gets the top-k similar users for user ui and recommends new services based on the score assigned to them. The kNN collaborative filtering suffers from the following: (1) It tends to favor recommending more popular items, potentially neglecting niche or personalized recommendations. (2) It captures only the static long-term preference of users, but it fails to analyze the dynamic preference of the users. (3) Computing the neighbourhood similarity for a large number of users is computationally slow. (4) As the algorithm needs to store all the training data, it is computationally expensive.

Neural Collaborative Filtering: Neural Collaborative Filtering (NCF) [6] uses a combination of Matrix Factorization (MF) and multi-layer perceptron (MLP). MF applies a linear kernel to model the latent feature interaction and MLP uses a non-linear kernel to learn the interaction from data. The model has a separate embedding layer for the MF and MLP because sharing the embedding between MF and MLP can limit its performance. Compared to the kNN collaborative filtering method, NCF exhibits improved performance. However, similar to its predecessors, NCF still falls short in capturing the dynamic preferences of users.

4.2. BERT4Rec

4.2.1. Architecture

The proposed Azure Service Recommendation model, using the BERT4Rec [7] achieves state-of-the-art performance. BERT4Rec utilizes the Self-Attention mechanism to understand the relevant portion of the input sequence. In each training sample, BERT4Rec analyzes the history of the sequence of services to identify the relevant services and uses them to predict the next service.

The sequence of services obtained in Figure 2 is applied as input to the model BERT4Rec as shown in Figure 3. A mask token is applied at the end of the sequence since the model needs to predict the next service the user is likely to add. Each service in the sequence is passed through the embedding layer M. As the Self-Attention model is not aware of the position, the learnable position embedding P is injected into the input embedding.

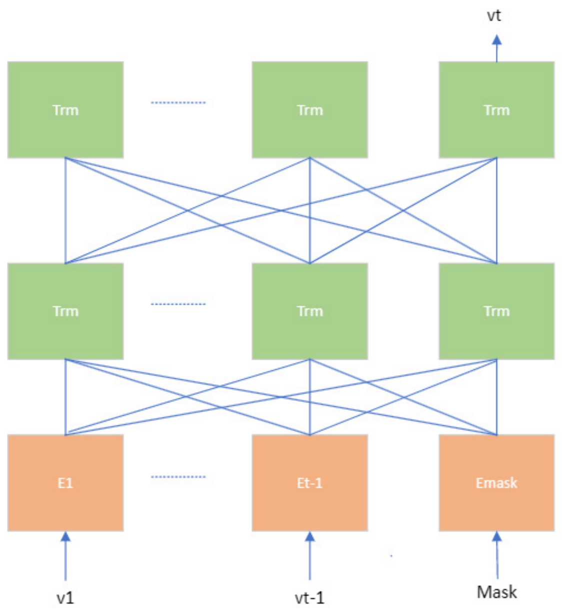


Figure 3: BERT4Rec architecture with 2 layers and embedding block. E_i represents the embedding and T_{rm} represents the Transformer block.

$$E = \begin{bmatrix} M_{s1} & + & P_1 \\ M_{s2} & + & P_2 \\ . & . & . \\ M_{sn} & + & P_n \end{bmatrix} \quad (1)$$

The Self-Attention layer SA accepts the embedding as input, multiplies it with weight of Query, Key and Value linearly and then passes it to the Attention layer.

$$S = SA(E) = Attention(EW^Q, EW^K, EW^V) \quad (2)$$

where E represents the embedding and W^Q, W^K, W^V represent the weight of Query, Key and Value respectively.

The Attention is expressed as follows:

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d}})V$$

where Q represent the queries, K the keys and V the values. The scale $\sqrt{d}$ avoids the large value of the dot product.

Self-Attention is a linear model. Non-linearity is required to learn complex dynamics. Thus, the output of the Attention layer is passed to the point-wise Feed-Forward Network. The point-wise Feed-Forward Network is expressed as follows:

$$F_i = FFN(S_i) = ReLU(S_i W^{(1)} + b^{(1)}) W^{(2)} + b^{(2)} \tag{3}$$

where $W^{(1)}, W^{(2)}, b^{(1)}, b^{(2)}$ represent the weight and bias and S_i represents the output of the attention layer.

The BERT4Rec consists of multiple blocks stacked on top of one another. Each block consists of both Self-Attention layers and Feed-Forward Network layers.

$$S^{(b)} = SA(F^{(b-1)})$$

$$F_i^{(b)} = FFN(S_i^{(b)})$$

The output of the model is given by:

$$r_{i,t} = F_t^{(b)} M_i^T \tag{4}$$

where $r_{i,t}$ is the relevance of service i being the next service given the first t services and M_i is the embedding output for the service i. $F_t^{(b)}$ can be represented as function depending on service embedding M.

$$F_t^{(b)} = f(M_{s1}, M_{s2}, ...M_{st})$$

4.2.2. Loss function

The Binary Cross-Entropy loss considers all the unused services as negative samples. Also, it can only consider implicit feedback and cannot directly use ratings for each of the services. Bayesian Personalized Ranking (BPR) [8] is a pairwise personalized ranking loss that is derived from the maximum posterior estimator. BPR loss prefers the used Services over the unused Services. As our model framework possesses a rating of services, we can leverage the rating using the BPR loss function where it will prefer a high-rated Service over a low-rated service.

The BPR loss is expressed as follows:

$$L = \sum_{(u,i,j \in D)} \ln \sigma(\hat{y_{ui}} - \hat{y_{uj}}) + \lambda_\Theta ||\Theta||^2$$

where $\hat{y_{ui}}$, $\hat{y_{uj}}$ are the output of the model for the user u, service i and service j where service i (positive sample) is preferred over service j (negative sample) and λ_Θ is the regularization parameter.

We compared the model trained using the BPR loss and Cross-Entropy loss and found that the model trained with BPR loss had a slightly higher Hit Ratio than Cross-Entropy loss.

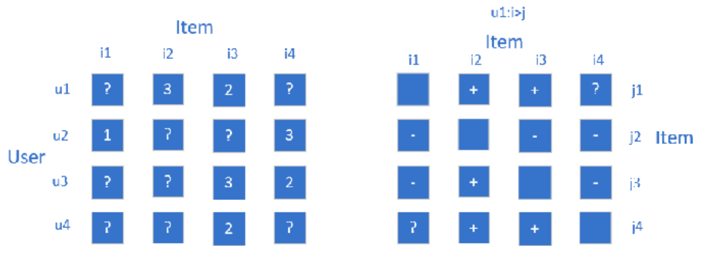


Figure 4: (a) The user–service matrix is filled with a corresponding rating.The ? indicates that the user has not used the service. (b) Indicates the pairwise preference for user u1, where the + sign indicates that user u prefers service i over service j and the - sign indicates that user u prefers service j over service i.

4.2.3. Cold start problem

The cold start problem arises when a new user joins the system, and the negligible data on user-service interactions makes it challenging to recommend services. To address this issue, we leverage the existing user features, such as Segment, Sub-segment, and Industry. These features prove valuable in recommending services to new users, as certain services are more commonly used within specific Segments, Sub-segments, and Industries. The embedding for each of the features is learned during model training. The output of the embedding layer for each feature is then concatenated into $E_{content}$ and added to the transformer layer output $F_t^{(b)}$, to make predictions for the next service.

$$E_{content} = Concat(E_{Seg}, E_{Subseg}, E_{Indus})$$

where E_{Seg}, E_{Subseg}, E_{Indus} represent the embedding of Segment, Sub-segment, Industry and $E_{content}$ represents the concatenated embedding. To address the cold start problem, we modify Eq. 4 as follows:

$$r_{i,t} = (F_t^{(b)} + E_{content}) M_i^T \tag{5}$$

4.2.4. Recommendation reason

Once BERT4Rec is trained, the service embedding of similar services exhibit high cosine similarity, as depicted in Figure 5. This occurs because similar services will have close representation in the latent space resulting in high cosine similarity.

After generating recommendations, we calculate the cosine similarity between each recommended service and all the services used by a particular user ID. We then list the top-2 used services in the recommendation reasons, as illustrated in Figure 6. Azure Marketing leverages the recommendation reason in the email campaigns and portal recommendations.

5. Result and analysis

In this section, we compare the performance of different models on our datasets.

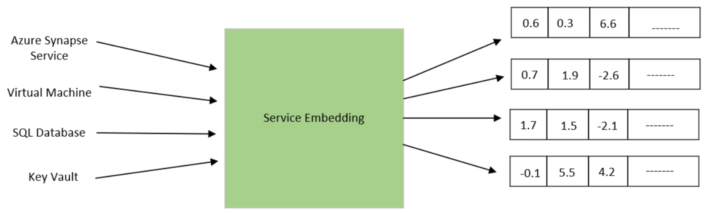
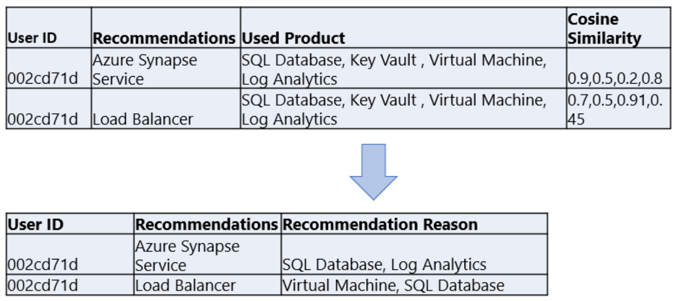


Figure 5: Service Embedding layer consisting of low dimensional latent space representation for each service.



User ID	Recommendations	Used Product	Cosine Similarity
002cd71d	Azure Synapse Service	SQL Database, Key Vault , Virtual Machine, Log Analytics	0.9,0.5,0.2,0.8
002cd71d	Load Balancer	SQL Database, Key Vault , Virtual Machine, Log Analytics	0.7,0.5,0.91,0.45

User ID	Recommendations	Recommendation Reason
002cd71d	Azure Synapse Service	SQL Database, Log Analytics
002cd71d	Load Balancer	Virtual Machine, SQL Database

Figure 6: Recommendation reason is obtained by the following: 1. Calculating the cosine similarity between the embedding of the recommended service and all the used products per user ID. 2. Picking the top-2 used products using the cosine similarity score as the recommendation reason.

5.1. Evaluation metrics

The performance of the model was evaluated using Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) **[6][9][10]**. A higher HR indicates that the recommendations made are highly relevant, as they are being adopted by the users. On the other hand, a higher NDCG ensures that relevant recommendations are ranked at the top of the recommendation list. In simpler terms, if relevant services are ranked lower, the NDCG score will be lower as well.

1. Hit Ratio (HR) is a metric defined as the ratio of the number of users ($|U^k hit|$) for whom the top-k recommendation is included in the target of the test dataset, to the total number of users ($|Uall|$) in the target of the test dataset. Mathematically, it can be expressed as follows:

$$HR@k = \frac{|U^k_{hit}|}{|U_{all}|}$$

2. Normalized Discounted Cumulative Gain (NDCG) is a metric that measures the ranking quality of recommendations. It is calculated as the ratio of the Discounted Cumulative Gain (DCG_k) to the Ideal Discounted Cumulative Gain ($IDCG_k$), where k represents the position in the recommendation list. The formulas for calculating DCG_k and $IDCG_k$ are as follows:

$$DCG_k = \sum_{i=1}^{k} \frac{rel_i}{log_2(i+1)}$$

$$IDCG_k = \sum_{i=1}^{REL_k} \frac{rel_i}{log_2(i+1)}$$

$$NDCG@k = \frac{DCG_k}{IDCG_k}$$

where rel_i is the relevance of the result at position i. The logarithmic reduction factor is used to penalize proportionally to the position of recommendation.

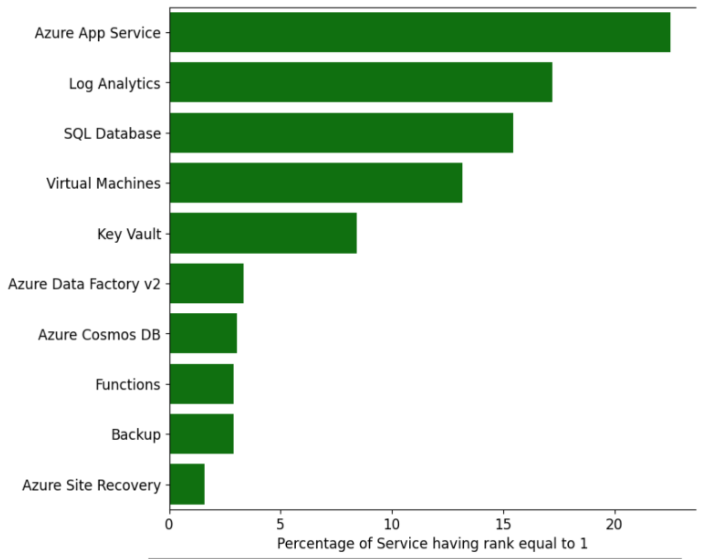
5.2. Comparison of model performance

We will compare the performance of the baseline model (kNN Collaborative filtering), NCF and BERT4Rec-based model. The time-based cross-validation approach was used for splitting the data into train and test periods. The model was trained using four months of data and the next four months of data was used for carrying out inference. It was found that the BERT4Rec-based model significantly outperforms kNN Collaborative Filtering as shown in Table 1. The BERT4Rec model output is required in batch mode for API calls from the Azure portal. The inference time for processing 2 million users is 1500 seconds, which translates to an average of 0.75 milliseconds per user.

Both the NCF-based and BERT4Rec-based models exhibit superior performance compared to kNN Collaborative Filtering. The NCF-based model achieved a 9% improvement in Hit Ratio and an 18% improvement in NDCG, while the BERT4Rec-based model demonstrated a remarkable 22% improvement in Hit Ratio and a substantial 31% improvement in NDCG.

5.3. Analysis of recommended service

We carry out the analysis of the top-recommended services by the BERT4Rec-based model. Figure 7 shows the distribution of services having rank = 1 and rank <= 5 respectively obtained after analyzing 2 Million user IDs. From Figure 7 (a), we can infer that Azure App Service, Log Analytics, and SQL Database are the top services having rank = 1. Similarly Figure 7 (b), shows that Key Vault, Log Analytics, and Azure App Service are the top services having rank <= 5.



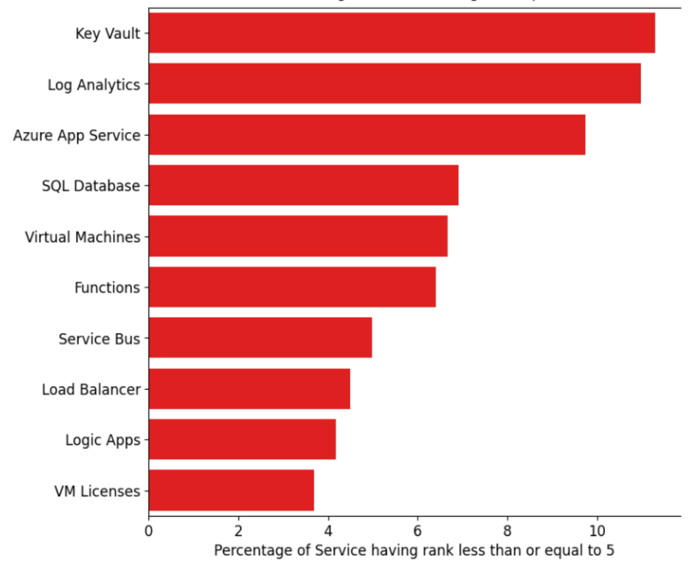


Figure 7: (a) Distribution of the Recommended Services having rank = 1, (b) Distribution of the Recommended Services having rank ≤ 5.

5.4. Ablation study

To examine the impact of hyperparameters on the model performance, an ablation study was conducted. Specifically, we focused on three key hyperparameters: the hidden dimension of the Attention layer, the dimension of the Feed-Forward layer, and the number of attention heads. The purpose of this study was to investigate how variations in these hyperparameters affect the overall performance of the model.

1. *Attention layer hidden dimension:* In order to investigate the impact of the attention layer dimension on the model performance, an ablation study was conducted. The model's performance was evaluated using HR@5 and NDCG@5 metrics. The dimension of the attention layers is determined by the weights of Query, Key, and Value (W^Q, W^K, and W^V). To explore the effect of different attention layer dimensions, the hidden dimension was varied from 16 to 64, as illustrated in Figure 8. The results indicated that a dimension of 64 yielded the best performance. Further increasing the hidden dimension beyond 64 did not significantly enhance the model's performance, but it did result in slower execution times due to the increased number of parameters.
2. *Feed-Forward layer dimension:* To examine the impact of the feed-forward layer dimension on the model performance, the dimension was varied from 8 to 64 as part of the ablation study. The model's performance was evaluated using relevant metrics. It was observed that both hidden unit values of 32 and 64 yielded similar performance.
3. *Number of heads:* We found out that increasing the number of heads in the Multi-head Attention layer improves model performance slightly, as shown in Figure 9. With the increase in the number of heads, we have multiple query, key and value matrices. Each of these sets projects into a different representation subspace, resulting in improved performance. The number of heads in the Multi-head Attention layer was varied from 1 to 4, and the model achieved its best performance with 4 heads.

6. Business impact

The Azure Service Recommendation engine has been successfully deployed across multiple marketing channels, reaching a wide range of users including Microsoft sellers, business planning teams, marketing departments, and direct customers. The valuable insights provided by the model has greatly benefited over 1000 sellers, resulting in an annual opportunity dollar value of 200M through MSX Daily Recommender/Sales Accelerator. Furthermore, the engine has played a crucial role in GDC's relationship management marketing campaign, where it achieved a remarkable 64% increase in Azure Synapse conversion rates during a recent experiment. Currently, the engine is undergoing User Acceptance Testing (UAT) for the purpose of surfacing direct customers on the Azure Portal, scheduled to be completed in June 2023.

7. Conclusion

In this research, we propose a novel BERT4Rec-based model for Azure Service Recommendation, which effectively addresses the sequential dynamics of user behavior.

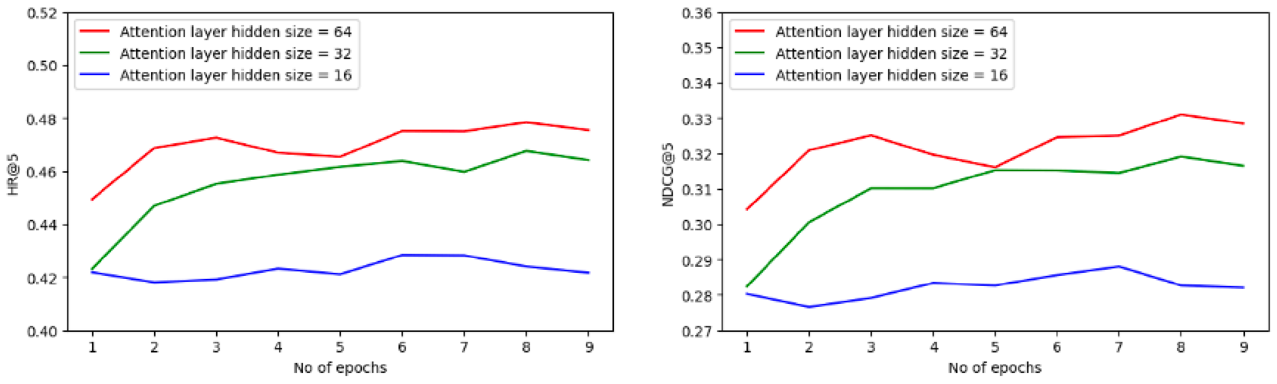


Figure 8: Impact of attention layer hidden dimension on model performance.

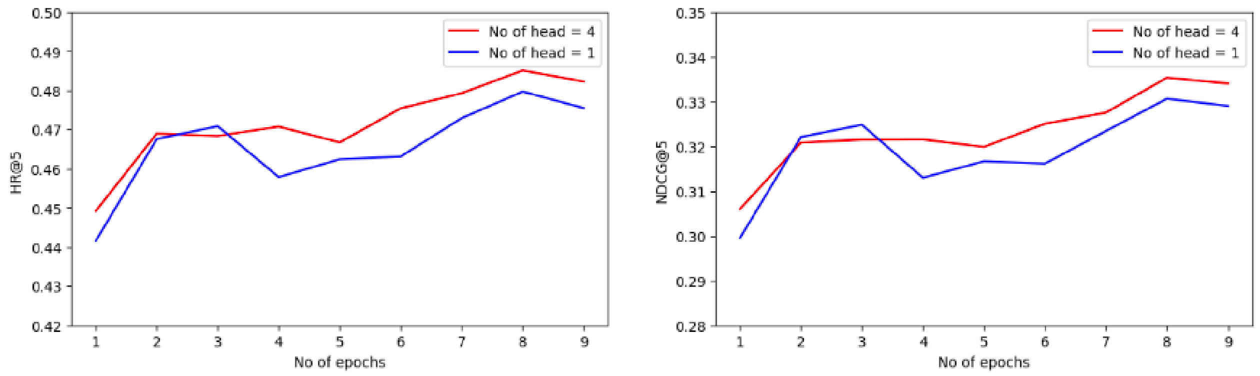


Figure 9: Impact of number of heads in multi-head attention layer on model performance.

Compared to the Nearest Neighbor Collaborative Filtering approach, our proposed BERT4Rec-based model achieves significant improvements, increasing the Hit Ratio by 22% and NDCG by 31%. Additionally, our model tackles the cold start problem by incorporating user features. The BERT4Rec-based model has been successfully deployed in a production environment and is currently being leveraged by the Marketing team for email campaigns and the Azure Portal. In the future, our research aims to conduct a performance comparison between BERT4Rec and the kNN model that incorporates sequence information. Specifically, we plan to explore two types of matrices for kNN: the user-time combination x service matrix and the user x service-time combination matrix. This approach will enable a more direct and meaningful comparison between the models as both models will be sequential, allowing for an apples-to-apples comparison.

In future email campaigns, we aim to analyze the revenue generated from each recommended service. Our model should not only prioritize the most probable next service but also recommend high-revenue services. By considering both relevance and revenue, we can increase the dollar revenue generated by the model.

For the Azure Portal, our future plan involves conducting A/B testing by equally splitting the traffic into control and treatment groups. The success metrics for evaluation will include the increase in the number of recommendation clicks and the Azure Consumption Revenue (ACR). Furthermore, we aim to enhance the model's performance by incorporating a feedback integration loop. This loop will integrate various information, such as the number of recommendation clicks, the position of clicked recommended services, search queries, and more. By incorporating this feedback, we expect to provide more accurate recommendations, resulting in further improvements in model performance.

8. Acknowledgements

We would like to thank our stakeholder team Joseph Kim from the Azure portal team, Melissa Kidwell and Urmila Dhole from the GDC RM team for helping us to understand the business requirement. We thank Lauren Brown, Program Manager 2 from CX Data for support during the project. Finally, we would thank Kiran R, Partner and Data Science Manager of the ML Science Team for his immense encouragement and support to do applied R&D in bringing business impact and publishing significant work.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkorei, Łukasz Kaiser, "Attention Is All You Need," 2017.

[2] Yehuda Koren, Robert Bell, Chris Volinsky, "Matrix Factorization Techniques For Recommender Systems," 2009.

[3] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, Lars Schmidt, "Factorizing Personalized Markov Chains For Next-Basket Recommendation," 2010.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, "BERT: Pre-Training Of Deep Bidirectional Transformers For Language Understanding," 2019.

[5] Alec Radford, JeffreyWu, Rewon Child, Rewon Child, David Luan, "Language Models Are Unsupervised Multitask Learners," 2019.

[6] Xiangnan He, Lizi Liao, Hanwang Zhang , "Neural Collaborative Filtering," 2017.

[7] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, "BERT4rec: Sequential Recommendation With Bidirectional Encoder Representations From Transformer," 2019.

[8] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, Lars Schmidt, "BPR: Bayesian Personalized Ranking From Implicit Feedback," 2009.

[9] Wang-Cheng Kang, Julian McAuley, "Self-Attentive Sequential Recommendation," 2018.

[10] Xiangnan He, Lizi Liao, Hanwang Zhang , "Fissa: Fusing Item Similarity Models With Self-Attention Networks For Sequential Recommendation," 2020.