data, it is difficult to accurately grasp the potential distribution of users' interests, which can lead to performance degradation.

To tackle these challenges, the focus has shifted towards generative models that are capable of learning the underlying data distribution and quantifying the associated uncertainties. These models aim to infer the probabilities of data generation by understanding the latent structures within the data. However, it's important to note that conventional generative models, such as Variational Autoencoders (VAEs) [4] and Generative Adversarial Networks (GANs) [5], often encounter issues like posterior collapse [6] or training instability, which can hinder their performance.

Therefore, influenced by the significant effectiveness of diffusion models in many research fields, it gradually adds noise to the raw data and iteratively denoise in reverse to reconstruct the data. This effectively alleviates the instability of one-step generative models and mitigates the problem of posterior collapse. Through this process, the diffusion model can effectively capture the potential interest distribution of users and effectively be used in sequential recommendation.

In DiffuRec [7], during the training phase, Gaussian noise is gradually added into the target item embedding, and the noisy target item embeddings obtained are combined with the user's interaction history data to serve as input for obtaining guidance on the target item. Then, the Transformer model is used as an approximator to learn the distribution of noise, capture users' interests, and reconstruct the target item.In the inference phase, the trained Transformer model is used for denoising, gradually predicting the target item from pure Gaussian noise data, and mapping it back to a discrete space through a rounding function to obtain the predicted target item.

This report reproduces the DiffuRec [7] model and conducts several ablation experiments. Building on this, and inspired by the EDiffuRec [8] article, we replaced the Gaussian distribution with a Weibull distribution to explore the effects of different noise distributions.

2 Related Work

2.1 Sequential Recommendation

Recommendation system technology has evolved from simple algorithms to complex models, and sequential recommendation methods are also constantly evolving. Early Markov chain models were used to simulate the transition probabilities of user's behavior, while modern methods utilize the powerful capabilities of deep neural networks to capture complex dependency relationships in sequential data. GRU4Rec [9] captures temporal dependencies through GRU, Caser [10] uses CNN to recognize patterns, SASRec [2] uses Transformer to simulate complex interactions, and BERT4Rec [3] combines Transformer and Cloze tasks to encode behavioral sequences. The development of these methods demonstrates their effectiveness and efficiency in behavior sequential model.

2.2 Diffusion Model

The diffusion model is a generative model that converts the data distribution into a simple prior distribution—a Gaussian distribution—by gradually adding noise, and then trains a model to reverse this process, restoring the original data distribution from the noise [11]. This process can be divided into two main stages: the forward process and the reverse process, as shown in Figure 1.

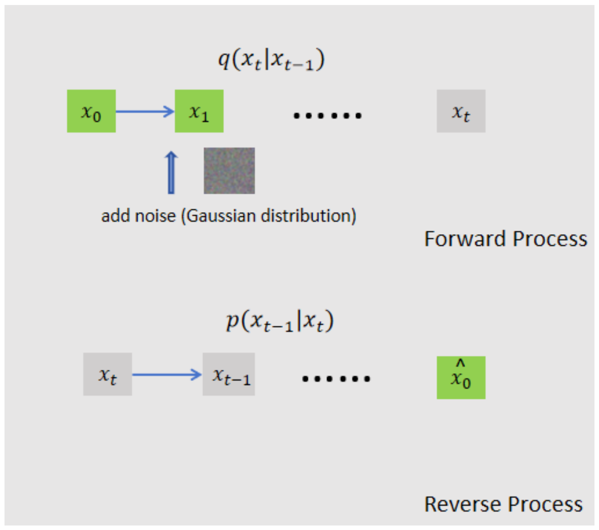

Figure 1: The process of diffusion.

Recently, it has been found that many studies use diffusion models for sequential recommendation. DiffRec [12] applies diffusion models to recommendation systems without considering temporal information, but its extension T-DiffRec [12] adopts a time-aware reweighting mechanism to improve the performance of the model. DiffuRec [7] uses Transformer to predict noise and reconstruct the target item. DreamRec [13] does not perform negative sampling, but directly replaces it with a generative paradigm. Diff4Rec [14], PDRec [15], and SeedRec [16] have achieved good results by employing diffusion models for data augmentation.

3 Method

This section provides a detailed explanation of the structure of DiffuRec and technical details used in DiffuRec.

In sequential recommendation, the dataset is represented as T. A set of user's historical interaction sequences as S_u, for each user, the interaction sequence is represented as $S_u = \{v_1^u, v_2^u, \ldots, v_n^u\}$, U represents the user set $U = \{u_1, u_2, \ldots, u_{|U|}\}$, V represents the item set $V = \{v_1, v_2, \ldots, v_{|V|}\}$. Generally speaking, each item $V \in T$ is converted into its corresponding embedding vector e_n. Therefore, the interaction sequence can be represented as $S_u = \{e_1, e_2, \ldots, e_n\}$.

3.1 Overview

To address the challenge of learning latent features amidst noise to extract useful information and bridge the gap between diffusion models and sequential recommendation, the article proposes DiffuRec, as depicted in Figure 2. DiffuRec is divided into three main components: (1) Diffusion Process: Gaussian noise is added to the target item for the approximator to learn; (2) Approximator: The Transformer learns the noise distribution and restores the original; (3) Reverse Process: Predicting the target item.

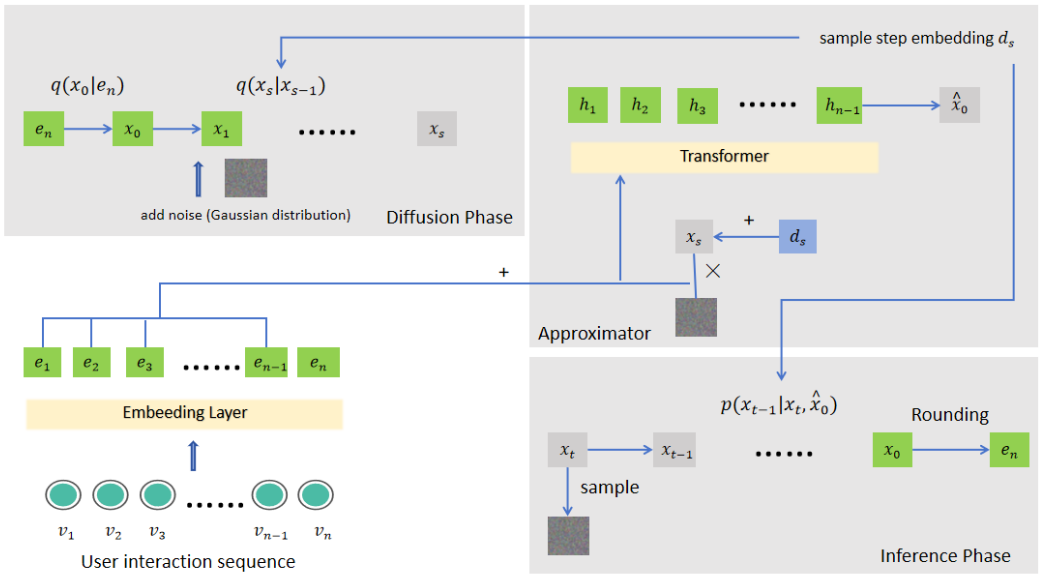

Figure 2: Architecture of DiffuRec.

Firstly, the user's historical interaction sequence S_u is embedded through the embedding layer to obtain the embedding representation e_n of the item. Then, Gaussian noise is gradually added to the embedding representation e_n of the target item, as detailed in section 3.2.1. Until a noisy representation x_t is obtained through step t, the timestep-embedding t from step t is sampled, and the noisy representation timestep-embedding is combined to introduce uncertainty parameters. Then, it is combined with the historical embedding vector to obtain the representation $Z = \{z_1, z_2, \ldots, z_{t-1}\}$, which is fed into the approximator to learn the reconstruction target item $\hat{x}_0$ and train to enhance the similarity between the reconstructed item and the target item embedding.

In the reverse (inference) phase, the noise data x_t from step t is sampled from the Gaussian distribution, and the representation $Z = \{z_1, z_2, \ldots, z_{t-1}\}$ obtained through the same processing as the diffusion process is fed into the trained transformer model to obtain the final representation $\hat{x}_0$. The model iterates from x_t to x_{t-1} until x_0, as detailed in section 3.2.2.

Finally, a rounding function is applied to map the reversed representation from the continuous space back to the discrete item space for target item prediction.

3.2 Diffusion Model

3.2.1 Forward Process

During the forward diffusion process, the raw data x_0 is transformed into noisy data x_t by gradually adding noise. This process can be expressed as:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{1}$$

Given the step size t, we can infer x_t from the initial x_0:

$$q\left(x_t|x_0\right) = \mathcal{N}\left(x_t; \sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)\mathrm{I}\right) \tag{2}$$

Among them, ϵ_t is the noise sampled from the standard normal distribution.$\beta_t = \{\beta_1, \beta_2, \ldots, \beta_t\}$ is variance schedule. α_t is a coefficient used to control the amount of noise added at each step, and it satisfies $\alpha_t = 1 - \beta_t$. $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. As time goes by, the features of data x_0 gradually disappear, and ultimately x_T approaches a pure noise distribution.

In DiffRec, the noise scheduling β_t for the discrete domain was explored using linear truncation [17], where a and b are hyperparameters and s is the current randomly sampled step, as shown in Equation 3.

$$\beta_s = \frac{a}{t}s + \frac{b}{s} \tag{3}$$

By using equations (2) and (3), the diffusion process of DiffuRec is completed, and the diffused state x_s is obtained.

3.2.2 Reverse Process

The goal of the reverse diffusion process is to recover the raw data x_0 from the noisy data x_T. This process is achieved by gradually denoising, with each step attempting to predict the noise data of the previous step x_{t-1}. This process can be expressed as [11]:

$$p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{4}$$

Rewrite the formula using Bayes' theorem as follows:

$$p_\theta\big(x_{t-1}|x_t, x_0\big) = N\big(x_{t-1}; \tilde{\mu}_t\big(x_t, x_0\big), \tilde{\beta}_t I\big) \tag{5}$$

$$\tilde{\mu}_t\left(x_t, x_0\right) = \frac{\sqrt{\bar{\alpha}_{t-1}}\left(1-\alpha_t\right)}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}\left(1-\bar{\alpha}_{t-1}\right)}{1-\bar{\alpha}_t}x_t \tag{6}$$

$$\tilde{\beta}_t = \frac{\Big(1-\alpha_t\Big)\Big(1-\bar{\alpha}_{t-1}\Big)}{1-\bar{\alpha}_t} \tag{7}$$

$\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$ are the mean and variance predicted by the neural network, respectively. The mean $\mu_\theta(x_t, t)$ and variance $\Sigma_\theta(x_t, t)$ during the reverse process are predicted by neural networks, which enables the model to adaptively denoise and generate latent variables that are closer to the raw data.

In the reverse process, x_0 is iteratively recovered from pure Gaussian noise x_t, as shown in equations (5) - (7). At this point, the neural network is a trained approximator (section 3.3).

3.3 Approximator

In DiffuRec, the approximator is a key component used for learning and simulating target item representation reconstruction. It is not only the core of learning data distribution and users' interests, but also the foundation for the model to make effective recommendations.

The main task of the approximator is to reconstruct the target item representation from the target item embedding with added noise during the diffusion phase. This is achieved by using a deep neural network (such as Transformer) that learns how to recover clear, noise free representations of target item from noisy data.

During the training phase, the approximator receives data processed through diffusion steps as input and attempts to predict the distribution of the raw data. At the same time, randomness (such as Gaussian noise) is introduced in the reconstruction process to simulate uncertainty and learn the complex distribution of data, and reconstruct the raw data. The process can be described as (The approximator is represented by $f_\theta(\cdot)$):

$$\mathbf{x}_s \leftarrow q(\mathbf{x}_s|\mathbf{x}_0, s) \tag{8}$$

$$\mathbf{z}_i = \mathbf{e}_i + \lambda_i \odot (\mathbf{x} + \mathbf{d}) \tag{9}$$

$$\hat{\mathbf{x}}_0 = f_\theta(Z_x) = \text{Transformer}\left([\mathrm{z}_0, \mathrm{z}_1, .., \mathrm{z}_{n-1}]\right) \tag{10}$$

Among them, d is the step embedding sampled during the diffusion or reverse process, x represents the noise state x_s of the target item embedding at this time, λ represents the uncertainty parameter introduced in the DiffuRec, which controls the mean and standard deviation of the generated Gaussian noise, thereby affecting the strength of the noise. Through this process, the model can simulate the uncertainty and diversity of users' interests, thereby enabling it to better adapt to changes in users' interests and improve the accuracy and robustness of recommendations.

In the inference phase, the approximator uses its learned knowledge to predict the next possible interaction item, starting from the user's historical interactions and working in reverse. This is achieved by iteratively recovering the target item representation from Gaussian noise.

$$\begin{aligned} &\hat{\mathbf{x}}_0 = f_\theta(\mathbf{Z}_{x_t}) \\ &\mathbf{x}_{t-1} \leftarrow p(\mathbf{x}_{t-1}|\hat{\mathbf{x}}_0, \mathbf{x}_t) \end{aligned} \tag{11}$$

3.4 Optimization

In diffusion models, since x_0 is unknown, deep neural networks are typically used to estimate x_0. The optimization objective is to minimize the following Variational Lower Limit (VLB):

$$\begin{aligned}\mathcal{L}_{vlb} = & \underbrace{E_q\left[D_{KL}\left(q(\mathrm{x}_t|\mathrm{x}_0)||p(\mathrm{x}_t)\right)\right]}_{L_t} \\ & + \underbrace{E_q\left[\sum_{s=2}^{t} D_{KL}(q(\mathbf{x_{s-1}}|\mathbf{x_s},\mathbf{x_0})||p_\theta(\mathbf{x_{s-1}}|\mathbf{x_s}))\right]}_{L_{t-1}} \\ & - \underbrace{\log p_\theta(\mathrm{x}_0|\mathrm{x}_1)}_{L_0}\end{aligned} \tag{12}$$

The L_t measures the KL divergence between the distribution of the diffusion data x_t generated from the raw data x_0 and the standard Gaussian distribution $p(x_t)$ at step t of the diffusion process. The L_{t-1} measures the sum of KL divergence between the distribution of the diffusion data x_{s-1} from the previous step and the model predicted distribution $p_\theta(x_{s-1}\ x_s)$ in the forward process (from step 2 to step s). L_0 is negative log-likelihood, which measures the accuracy of the model's prediction of the raw data x_0 given the diffusion data x_1 in the first step of the reverse process.

However, VLB loss can easily lead to unstable model training, so the mean square error loss can be used to minimize the predicted $\hat{x}_0$ The difference between the embedding of $\hat{x}_0$ and the real target item can be simplified as [11]:

$$\mathcal{L}_{simple} = E_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[||\boldsymbol{\epsilon} - \boldsymbol{\epsilon_\theta}(\sqrt{\overline{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\boldsymbol{\epsilon}, t)||^2\right] \tag{13}$$

$\boldsymbol{\epsilon_\theta}$ representing approximators.

However, in the specific implementation of DiffuRec, in order to simplify the problem and adapt to the characteristics of the recommendation system, the author chose cross entropy loss to measure the difference between the target item representation predicted by the model and the true target item. Refer to formulas (14) and (15):

$$\hat{y} = \frac{\exp(\hat{\mathrm{x}}_0 \cdot \mathrm{e}_{n+1})}{\sum_{i\in I}\exp(\hat{\mathrm{x}}_0 \cdot \mathrm{e}_i)} \tag{14}$$

$$\mathcal{L}_{CE} = \frac{1}{|\mathcal{U}|}\sum_{i\in\mathcal{U}} -\log\hat{y}_i \tag{15}$$

In practical applications, this loss function is more suitable for the goal of recommendation systems, which is to predict the next item that the user may be interested in given the user's historical behavior sequence. By minimizing cross-entropy loss, the model can learn how to adjust the probability predictions for different items based on the user's historical behavior, thereby improving the accuracy of recommendation.

Finally, the reversed representation in the continuous space is mapped back to the discrete item space using a rounding function.

$$\arg\max_{i\in I} \quad \mathrm{Rounding}(\mathbf{x_0}) = \mathbf{x_0}\cdot\mathbf{e_i}^T \tag{16}$$

4 Implementation Details

4.1 Comparison with existing open source code

This project refers to the source code of https://github.com/WHUIR/DiffuRec as the basic framework, and this project modifies and improves it to make it work properly, and to a certain extent.

This project is inspired by the EDiffuRec framework (EDiffRec is not open source), which suggests that the noise distribution can be replaced with other effective distributions. On this basis, this report attempts to use Weibull distribution instead of Gaussian distribution in DiffuRec for comparative experiments and analysis.

4.2 Experimental environment setup

4.2.1 Dataset

The experiment was conducted using four benchmark datasets: Beauty, Toys, Moviellens-1M, and Steam. The statistical data of these datasets are shown in Table 1.

Table 1: Dataset statistical information

Dataset	user	item	Interaction
Beauty	22363	12101	198502
Toys	19412	11924	167597
Movielens-1M	6040	3416	999611
Steam	281428	13044	3485022

4.2.2 Experimental details

The experiment was conducted in the Pytorch 1.10.1 and CUDA 11.1 environments. The data is sorted by timestamp and inactive users and unpopular items with fewer than 5 interactions are removed. Using Adam optimizer, the initial learning rate is 0.001. The parameter initialization of the Transformer basic model adopts Xavier normalized distribution. The embedding dimension and hidden state size in the model are set to 128, and the batch size is 512. The dropout rate is 0.1. For the noise parameter θ, sample from a Gaussian distribution with a mean of 0 and a variance of 0.001. The total number of reverse steps was set to 32, and the impact of different reverse steps on model performance was analyzed in the experiment. We chose truncated linear scheduling as the noise scheduling strategies and compared it with other scheduling schemes such as square root, cosine, and linear in the experiment.

4.3 Innovation

This project is inspired by the EDiffuRec framework (EDiffRec is not open source), which suggests that the noise distribution can be replaced with other effective distributions.

The attempt to replace the distribution is:

1. The distribution learned by the approximator remains unchanged, and the noise distribution is replaced with a Weibull distribution in the forward diffusion process;
2. The distributions learned by diffusion forward process and reverse process are all replaced with Weibull distribution.

Only the second point was discussed in the EDiffuRec article, whereas the first point is based on my personal attempt.

5 Result and Analysis

Table 2 shows the experimental results, and Table 3 shows the results of the ablation experiment. It can be found that the reproduced experimental results are basically consistent with the results in the article.

Table 2: Comparison of Results(%)

Dataset	Metric	DiffuRec(article)	DiffuRec(Project)
Beauty	HR@5	5.5758	5.0899
	HR@10	7.9068	7.3646
	HR@20	11.1098	10.4174
	NDCG@5	4.0047	3.608
	NDCG@10	4.7494	4.3405
	NDCG@20	5.5566	5.1075
Toys	HR@5	5.565	5.7034
	HR@10	7.4587	7.4667
	HR@20	9.8417	9.8192
	NDCG@5	4.1667	4.2462
	NDCG@10	4.7724	4.8152
	NDCG@20	5.3684	5.4077
Movielens-1M	HR@5	17.9659	15.6585
	HR@10	26.2647	23.9417
	HR@20	36.787	35.3465
	NDCG@5	12.115	10.376
	NDCG@10	14.7909	13.0424
	NDCG@20	17.4386	15.9141
Steam	HR@5	6.6742	6.6417
	HR@10	10.752	10.7433
	HR@20	16.6507	16.6189
	NDCG@5	4.2902	4.2742
	NDCG@10	5.5981	5.5892
	NDCG@20	7.081	7.0656

Ablation experiment:

- w GRU:The approximator structure has been changed from Transformer to GRU.
- w Cosine:Change the noise strategy to cosine schedule.

- w L:Change the noise strategy to a linear schedule.
- w sqrt:Change the noise strategy to a square root schedule

Table 3: Comparison of ablation experiment results(%) on the Beauty dataset

Method		HR@5	HR@10	HR@20	NDCG@5	NDCG@10	NDCG@20
DiffuRec(article)	w GRU	3.1773	4.6685	6.9000	2.2253	2.7075	3.2682
	w Cosine	5.2502	7.2967	10.2858	3.7908	4.4461	5.1974
	w L	5.1420	7.3428	10.2650	3.7538	4.4610	5.1965
	w Sqrt	5.1543	7.2871	10.4156	3.7600	4.4443	5.2288
	DiffuRec	5.5758	7.9068	11.1098	4.0047	4.7494	5.5566
DiffuRec(Project)	w GRU	3.3495	4.7779	6.9851	2.3727	2.833	3.3897
	w Cosine	5.6726	7.8482	10.7217	4.0924	4.7916	5.5111
	w L	5.5976	7.8087	10.8118	4.0591	4.7731	5.5287
	w Sqrt	5.75	7.9078	10.6664	4.1409	4.8307	5.5229
	DiffuRec	5.0899	7.3646	10.4174	3.608	4.3405	5.1075

Table 4 presents the improved method and comparative results.

- DiffuRec (Weibull) and DiffuRec (Weibull-1) represent the results of replacing the distributions learned by the noising process and denoising process with Weibull distributions.
- DiffuRec (Weibull-1) * represents replacing the noising process with a Weibull distribution, and the distribution learned by the Transformer remains the result of Gaussian noise.

The difference between DiffuRec (Weibull) and DiffuRec (Weibull-1) is that $c = 0$ or $c = -1$. The Weibull distribution follows:

$$f(x) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} + c, & x \geq c, \\ 0, & x < c. \end{cases} \tag{17}$$

Table 4: The results(%) of improving and comparing the DiffuRec

Dataset		DiffuRec(article)	DiffuRec(Project)			
			DiffuRec(Project)	DiffuRec(Weibull)	DiffuRec(Weibull-1)	DiffuRec(Weibull-1)*
Beauty	HR@5	**5.5758**	5.0899	0.0621	0.0755	5.4691
	HR@10	**7.9068**	7.3646	0.1131	0.1332	7.8324
	HR@20	**11.1098**	10.4174	0.2241	0.2486	10.8302
	NDCG@5	**4.0047**	3.608	0.0376	0.0468	3.9743
	NDCG@10	**4.7494**	4.3405	0.0543	0.0657	4.7317
	NDCG@20	**5.5566**	5.1075	0.0819	0.0948	5.4866
Toys	HR@5	5.565	5.7034	0.0622	0.5859	**5.7654**
	HR@10	7.4587	7.4667	0.093	0.9354	**7.5689**
	HR@20	9.8417	9.8192	0.1855	1.4905	**10.0299**
	NDCG@5	4.1667	4.2462	0.0336	0.3989	**4.3163**
	NDCG@10	4.7724	4.8152	0.0435	0.5112	**4.895**
	NDCG@20	5.3684	5.4077	0.0661	0.6502	**5.512**

The parameters k and λ that determine the shape and scale are fixed at 2.0 and 0.5, while the position parameter c allows for different ranges of variation and determines the starting point of the distribution.

The results in Table 4 indicate that the performance of the diffusion model is closely related to the noise distribution used. The model needs to be optimized and adjusted for specific noise distributions to ensure optimal performance. In addition, the generalization ability of the model and its adaptability to different noise distributions are also important factors affecting performance.

The results of mixed noise distribution show that the performance under this mixed setting is similar to that of the original Gaussian distribution, indicating that the Transformer has a certain robustness to noise distribution and can adapt to different noise distributions to a certain extent.

The significant difference in the effect after replacing the noise distribution may be caused by various factors, and different datasets have different characteristics, including user behavior patterns, item distribution, interaction frequency, etc. Gaussian distribution and Weibull distribution have different shapes and characteristics, and may be more suitable for simulating noise or uncertainty in certain datasets. If the characteristics of Weibull distribution do not match the noise patterns in the dataset, it may lead to a decrease in model performance.

The Transformer may have certain prior assumptions about Gaussian noise during the design and training process. These assumptions may not fully match the characteristics of the Weibull distribution, leading to a decrease in the efficiency of the model when dealing with Weibull noise. And Gaussian distribution and Weibull distribution have different tail behaviors. The tail decay rate of Gaussian distribution is slower, while the tail of Weibull distribution may be heavier or lighter, which may affect the model's ability to handle extreme or rare events. After replacing the noise distribution, the model may also face the risk of overfitting or underfitting. If the model overfits on a new noise distribution, it may perform poorly on the test set; If the model is underfitting, it may not be able to capture key information in the data.

6 Conclusion and Future work

The experimental results demonstrate that DiffuRec has achieved significant performance improvements on four real-world datasets, outperforming existing baseline methods for sequential recommendation. This underscores the potential application of diffusion models in sequential recommendation. DiffuRec's success lies in its ability to not only better capture the dynamics of user interests and the multidimensionality of item characteristics but also to enhance the diversity and novelty of recommendation systems by introducing uncertainty. Moreover, DiffuRec's design takes into account the guiding role of target items, which helps the model more accurately understand user intentions, thereby improving the relevance and accuracy of recommendations. Overall, DiffuRec offers a new perspective and approach in the field of sequential recommendation, showcasing the powerful potential of diffusion models in grasping complex user behaviors and item features.

Future efforts can concentrate on delving deeper into the potential of the hybrid approach, and endeavor to elucidate the reasons behind its notable effectiveness within particular scenarios.

References

[1] A Vaswani. "Attention is all you need". In: *Advances in Neural Information Processing Systems* (2017).

[2] Wang-Cheng Kang and Julian McAuley. "Self-attentive sequential recommendation". In: *2018 IEEE international conference on data mining (ICDM)*. IEEE. 2018, pp. 197–206.

[3] Fei Sun et al. "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer". In: *Proceedings of the 28th ACM international conference on information and knowledge management*. 2019, pp. 1441–1450.

[4] Dawen Liang et al. "Variational autoencoders for collaborative filtering". In: *Proceedings of the 2018 world wide web conference*. 2018, pp. 689–698.

[5] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[6] James Lucas et al. "Understanding Posterior Collapse in Generative Latent Variable Models". In: *Available online: https://openreview.net/forum?id=r1xaVLUYuE*. 2019.

[7] Zihao Li, Aixin Sun, and Chenliang Li. "Diffurec: A diffusion model for sequential recommendation". In: *ACM Transactions on Information Systems* 42.3 (2023), pp. 1–28.

[8] Hanbyul Lee and Junghyun Kim. "EDiffuRec: An Enhanced Diffusion Model for Sequential Recommendation". In: *Mathematics* 12.12 (2024), p. 1795.

[9] B Hidasi. "Session-based Recommendations with Recurrent Neural Networks". In: *arXiv preprint arXiv:1511.06939* (2015).

[10] Jiaxi Tang and Ke Wang. "Personalized top-n sequential recommendation via convolutional sequence embedding". In: *Proceedings of the eleventh ACM international conference on web search and data mining*. 2018, pp. 565–573.

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[12] Wenjie Wang et al. "Diffusion recommender model". In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2023, pp. 832–841.

[13] Zhengyi Yang et al. "Generate what you prefer: Reshaping sequential recommendation via guided diffusion". In: *Advances in Neural Information Processing Systems* 36 (2024).

[14] Zihao Wu et al. "Diff4rec: Sequential recommendation with curriculum-scheduled diffusion augmentation". In: *Proceedings of the 31st ACM International Conference on Multimedia*. 2023, pp. 9329–9335.

[15] Haokai Ma et al. "Plug-in diffusion model for sequential recommendation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 8. 2024, pp. 8886–8894.

[16] Haokai Ma et al. “Seedrec: sememe-based diffusion for sequential recommendation”. In: *Proceedings of IJCAI*. 2024, pp. 1–9.

[17] Emiel Hoogeboom et al. “Argmax flows and multinomial diffusion: Towards non-autoregressive language models”. In: *arXiv preprint arXiv:2102.05379* 3.4 (2021).