Two-sided Rank Consistent Ordinal Regression for Interpretable Music Key Recommendation

Yuan Wang
ywang4@scu.edu
Santa Clara University
USA

Shigeki Tanaka
shigeki.tanaka.da@nttdocomo.com
NTT DOCOMO, INC.
Japan

Keita Yokoyama
keita.yokoyama.sh@nttdocomo.com
NTT DOCOMO, INC.
Japan

Hsin-Tai Wu
hwu@docomoinnovations.com
DOCOMO Innovations, Inc.
USA

Yi Fang
yfang@scu.edu
Santa Clara University
USA

ABSTRACT

Model interpretability has attracted increasing attention in the IR community since it is important to ensure that end-users (decision-makers) correctly understand and consequently trust the functionality of the models. On the other hand, ordinal regression has been widely used in many ranking and prediction tasks, but it could not guarantee the rank consistent predictions for the output labels, which makes the predicted results hard to explain. Take the music key recommendation in karaoke as an example where a user could select a key ranging from -7 to +7 so that the song could meet the user's vocal competence for better performance. If the best key for a user to sing a song is -3, the keys smaller than -3 should be ranked in decreasing order. Similarly, the keys on the positive side should also be ranked in the decreasing order. To address this challenge, we propose a novel Two-sided Rank Consistent Ordinal Regression model. We show that the model is not only able to predict the key for the target song given the user's singing history, but it also has the theoretical guarantees for the two-sided rank-monotonicity. We train the model with a history encoder using the recurrent units and a key decoder using the Transformer. The experimental results on the real-world karaoke dataset demonstrate the effectiveness of our proposed model.

CCS CONCEPTS

• **Information systems** → **Music retrieval**.

KEYWORDS

Ordinal regression, Model interpretability, Music information retrieval

ACM Reference Format:
Yuan Wang, Shigeki Tanaka, Keita Yokoyama, Hsin-Tai Wu, and Yi Fang. 2022. Two-sided Rank Consistent Ordinal Regression for Interpretable Music Key Recommendation. In *Proceedings of the 2022 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '22), July 11–12, 2022, Madrid, Spain.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3539813.3545147

ICTIR '22, July 11–12, 2022, Madrid, Spain

ACM ISBN 978-1-4503-9412-3/22/07…$15.00
https://doi.org/10.1145/3539813.3545147

1 INTRODUCTION

Interpretable IR and machine learning attempt to develop models that generate not only high-quality predictions but also intuitive explanations. In this way, it helps to improve the transparency, persuasiveness, trustworthiness, and user satisfaction of the systems. It also facilitates system designers to diagnose, debug, and refine the algorithm. On the other hand, ordinal regression is an important technique in the IR and machine learning communities as it can take the ordering information between the target labels into consideration. Ordinal regression has been widely used in many ranking and prediction tasks [5, 11, 38]. The ordinal task is usually transformed into multiple binary classification tasks using the binary cross-entropy loss. However, these approaches could not guarantee the monotonicity among the predicted labels, which makes them hard to explain. Intuitively, the ordinal model should retain the ordering information in the predicted outputs such that the predicted probability for each ordinal label should be in an increasing or decreasing order. For example, in the age estimation task, if the ground truth label is 60, the labels such as 50, 40, and 30 should have monotonically decreasing probabilities. On top of the aforementioned rank consistent output, the two-sided rank consistent output is a more challenging problem. In some cases, the ordering information among the ordinal labels is not only increasing and decreasing, but the ordering could be increasing then decreasing, which we called the two-sided monotonicity. Under this setting, the desired predicted probability from the model should also be two-sided monotonic.

One real-world application of the two-sided monotonicity is the music key recommendation in karaoke. Karaoke machines are becoming an increasingly popular choice for many people's daily entertainment. After the user sings a song, the machine's rating system provides a real-time and automatic evaluation of the user's singing record. Besides dedicated practicing, one of the simple techniques to deliver better performance is to select a suitable key so that the song could meet the users' vocal competence. In the karaoke system, 15 keys are ranging from -7 to 7, with the default key being 0. In music, a key is a scale around which a piece of music revolves, and each key has the seven notes which make up

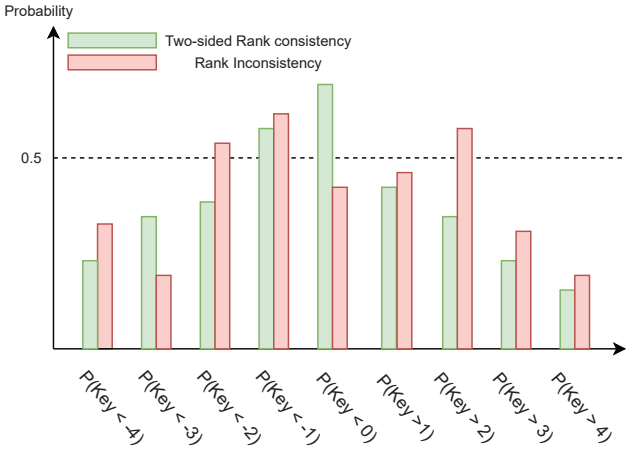


Figure 1: Illustration of the ideal probability of the model output. The two-sided rank consistency should be maintained in the model output for better interpretability where the probabilities increase and decrease consistently as the value of key increases and decreases.

the song's melody. For example, in the karaoke machine, using the positive keys will increase the pitch value and vice versa. As each individual has different vocal capabilities, using the default key may not be the best choice. The default key is specially chosen to fit everyone's vocal competence, but it is not the best choice for the user to get the highest score. The user needs to choose a suitable key to sing, which could lead to better performance. However, for most of the users who are not an expert with musical knowledge, they are not aware of the benefit of changing keys when the default key is not fitted. Thus, helping these users decide the best key for the target song becomes an important task.

In practice, the user has to sing at least one song before we could help him or her choose a key for the next target song. That is, without any historical records, it is impossible to recommend a key for a user to sing. Moreover, since everyone has a unique vocal key range, a personalized recommendation system is required. In collaboration with a Japanese karaoke company, we are provided the user's singing records and the vocal track of songs in the Midi format. Midi format is commonly used when composing music, and there is a lot of music dataset in this format such as The Lakh MIDI dataset [28]. In this format, the note's notation, pitch, velocity, and duration are all recorded.

In this key recommendation task, to provide reasonable and interpretable results, the relative ordering information between keys needs to be captured, which means the task should be formulated as the ordinal problem. For example, if the best key for a user to sing a song is -3, the keys smaller than -3 should be ranked in decreasing order. Also, the keys in the range 0 to +7 should be ranked in the decreasing order as these keys have increasing key values. As common sense, the song becomes more difficult to sing as the key is closer to -7 or +7 because the resulting song requires the users to have a very high or low vocal pitch range. This two-sided rank-monotonicity needs to be guaranteed because it is contradictory if the best key is -3 and the second-best key is +5. The desired output should be in the consistent two-sided rank. Figure 1 illustrates inconsistent (in red) and consistent (in blue) predictions respectively. If we use the traditional classification loss functions such as multi-category cross-entropy, we could not consider the rank-monotonicity between keys. Although the neural network-based ordinal regression [26] takes the ordering information into account, it suffers from the inconsistency among the label rankings. To tackle this problem, we propose the two-sided Rank Consistent Ordinal Regression (TRCOR) model for the key recommendation task in this paper. To the best of our knowledge, this paper presents the first study of the two-sided rank-monotonicity among the class labels with ordering information. Our main contribution can be summarized as follows:

- We propose a novel Two-sided Rank Consistent Ordinal Regression (TRCOR) model with theoretical guarantees for two-sided rank-monotonicity.
- We implement the music key recommendation model which can analyze the user's history singing records and accurately predict the target key with interpretable output.
- Experiments on the real-world karaoke dataset show the effectiveness of the proposed model.

2 RELATED WORK

2.1 Model Interpretability

Interpretability in machine learning has been extensively investigated in classical machine learning. Deep learning models have largely improved performance, but they tend to be opaque and less interpretable. As a result, the interpretability of these complex models has been studied in various domains such as image classification [9], sequence to sequence modeling [3], and named entity recognition [2], to better understand decisions made by the models. In IR, there has been relatively limited work on model interpretability [37]. A recent showed how a model introspective method meant for computer vision can be applied to interpreting the relevance score of a single query document pair [12]. The explainability of recommendation systems has also received increasing attention in recent years. An overview of the field can be found in [44]. To the best of our knowledge, no prior work has addressed the interpretability of music key recommendation.

2.2 Ordinal Regression

Ordinal regression is widely used in the classification problem where the class labels have relative ordering information. In the machine learning field, the ordinal regression problem was firstly reformatted to utilize multiple binary classification tasks [23]. There are also other works using perceptrons [8, 36] and support vector machines [7, 18, 29, 35]. Recently, some convolutional neural networks were proposed on age classification [22, 32], as the age estimation problem perfectly fits in the setting of ordinal regression. Also, a neural network-based framework was proposed to reduce the ordinal regression to the binary classification problem, which was also applied to the age classification problem by [26]. The proposed Ordinal Regression CNN (OR-CNN) transforms the K ranks

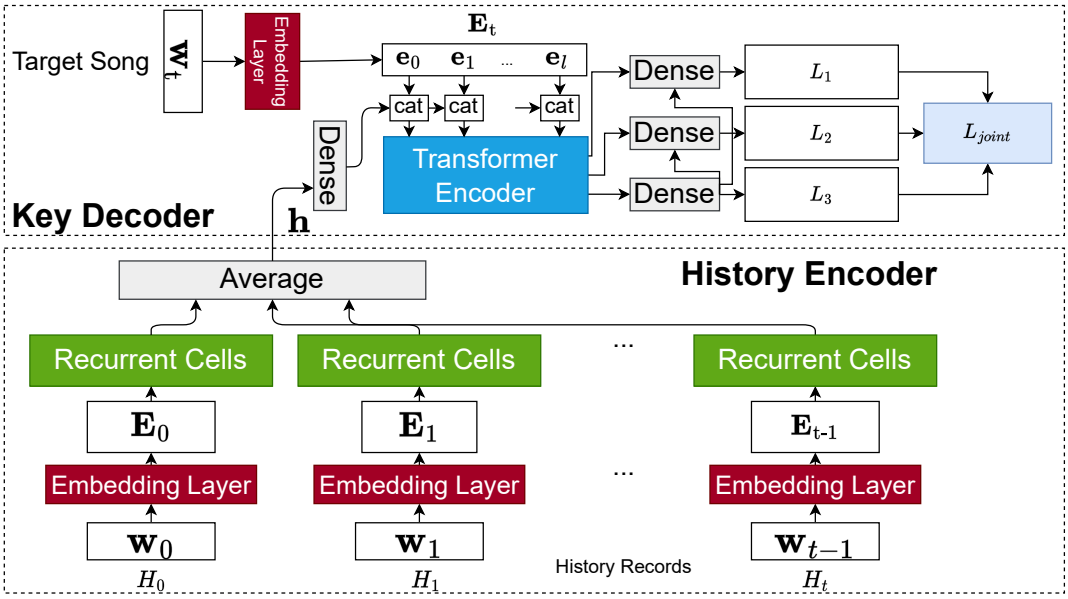


Figure 2: Architecture of the proposed TRCOR model. Here L_1 and L_2 denote the two rank consistent loss functions, which optimize the output probabilities for key from the upper range (0 to +7) and the lower range (-7 to 0) respectively. The rank consistent loss functions guarantee a consistent decreasing output probabilities for each of the key ranges, which lead to a two-sided rank consistency for all keys.

into $K-1$ binary classification problems, and all $K-1$ tasks share the same weights for the layers in the neural networks. However, the inconsistency between the predicted labels becomes a problem, which is solved by CORAL framework [6]. The CORAL framework could guarantee the rank consistency for the ordered labels, but it could not be used in our task which requires the two-sided rank-monotonicity. Moreover, for personalized rating prediction, since the feedback ratings could reflect the users' intention on the items, and the ratings are on the ordinal scale, the ordinal regression is used in the proposed recommendation model [21].

2.3 Karaoke Recommendation

Traditional recommendation system models such as Factorization Machines [30], BPR [31], and collaborative filtering [33] may not fit our problem because we are not recommending items such as movies and songs. In the existing work, some research have been conducted on music and karaoke song recommendation. In music recommendation systems, Downie [10] proposed a metadata-based model which uses textual metadata. Also, models using collaborative filtering techniques are widely used [1, 4, 41]. Recently, as deep learning becomes popular, T-recsys [13], the hybrid model [42], and the deep content-based model [39] are proposed to adapt the power of neural networks. Different from the music recommendation, the karaoke song recommendation aims to recommend the songs that would fit the users' vocal competence instead of their taste preferences. MKLA [15] is a karaoke recommender system that could compute the matching degree of users and songs using multiple kernel learning. InfoFuMF [17] used the matrix factorization to learn the user and song's latent feature vectors from the singing records, ratings, user data, and song data. CBNTF [16] proposes a joint model using negative tensor factorization and a support vector machine which also learns from the singing recordings. In [24], the authors design a model to acquire singer profiles from the vocal range profile which could be used as features for the recommendation models. However, all models mentioned above only aim to recommend the song that fits the users' vocal competence and ignore that the key is also an important factor during the recommendation. A recent work [43] addressed a karaoke key recommendation task, but it did not consider the two-sided rank monotonicity, which made the results hard to digest for end users.

3 DATA DESCRIPTION

In collaboration with Daiichikosho, one of the largest producers of karaoke machinery in the world, we are provided the dataset. In this karaoke dataset, there are the users' history singing records $\mathbf{H}$ and the notes for songs' vocal tracks $\mathbf{N}$. There are a total of 602,587 records and 20,914 users. The best key for each song and user is not given since the users could only know their performance ratings afterward. It is also rare for a user to sing a song for all 15 keys to find a suitable key. Here the performance ratings range from 0 to 1, which are calculated by dividing the number of correctly sung notes by the total number of notes. Thus, we apply a strict rating threshold of 0.95 so that we consider these records to be the pairs of users and songs with the best keys. By default, the records with key 0 have the most cases, so we down-sample the key 0 to make a balanced dataset.

For each s^{th} song in $\mathbf{N}$, $\mathbf{n}_s$ is a sequence of pairs that contains the note number and duration. Specifically, the sequence of notes

contains only the notes for the vocal track of the songs. For each singing record in $\mathbf{H}$ for a user, there is the sequence of pairs $\mathbf{r}_t$ that contains the note and duration sang by the user. We use i to denote the i^{th} user from the whole dataset. Also, each user has T history records where $T = |\mathbf{H}^i|$. Note that the duration of the notes is predetermined by the karaoke machine rating system.

4 PROPOSED MODEL

In this section, we describe our proposed TRCOR model for the key recommendation. Our method aims to predict the best key for the user to sing a song based on the history singing records. Our setting assumes the user has had at least one record. To address the problem of the two-sided rank-monotonicity, we first propose a model in the encoder-decoder structure consisting of a history encoder and a key decoder, and then we use three loss functions including two rank consistent losses and one binary cross-entropy (BCE) loss to achieve our goal. Each output probability for the keys from the upper range (0 to +7) and the keys from the lower range (-7 to 0) is optimized independently by the rank consistent loss respectively. For the two rank consistent loss functions, we adapt the setting from the CORAL [6] framework which could guarantee consistent decreasing output probabilities. The BCE loss could determine which range the predicted key belongs to. Since the BCE loss and the rank consistent losses have a different scale, we compute the weighted sum of the values from these losses so that we could optimize them jointly.

Within the TRCOR model, at each time t, the user u with history singing records $\mathbf{H}^i$ selects a target song with notes n_{t}. We first input the history singing records $\mathbf{H}^i = \{\mathbf{H}_0, \mathbf{H}_2, ...\mathbf{H}_{T-1}\}$ into the history encoder to get a history vector $\mathbf{h}_T$. Then the $\mathbf{h}_T$ and n_{t} are input into the key decoder to compute the probability for each keys.

4.1 Preprocessing

Since the original note sequences only contain the discrete value of notes, we need an informative representation for the songs and users' records so that the note, the duration, and the key could all be encoded together. Inspired by Word2Vec [25] originally proposed for word representation learning, we combine the notes and duration information into word strings and then apply the embedding layer in the model to learn an end-to-end representation for notes.

To combine the notes and duration information, we first determine a standard minimum time interval and compute the number of the time interval in each note duration. Then we convert each note integer into a string by repeating the integer by the number of intervals. For example, given a tuple $\{45, 0.005\}$ where 45 is the note and 0.005 is the note duration, we convert it into the string '45_45_45_45_45' if we set the standard time interval to be 0.001. Thus, each note sequences $\mathbf{n}_{\text{s}}$ could be converted into words sequences $\mathbf{w}_s = \{w_1, ..., w_{|\mathbf{w}_s|}\}$. Then using the embedding layer, each word in the sequence $\mathbf{w}_j = \{w_1, ..., w_{|\mathbf{w}_s|}\}$ is encoded as $\mathbf{e} = \Phi(w)$ where $\mathbf{e}$ is the d-dimensional embedding and Φ is the embedding layer. Here the embedding layer is a look-up table which stores the embedding vector for all words in our note words dictionary. The vectors in the embedding layer are updated along with other parameters while we train the model. Thus, the input to the model could be denoted as $\mathbf{x}_i = \{\mathbf{H}^i_0, ..., \mathbf{H}^i_{t-1}, \mathbf{E}_t\}$.

4.2 History Encoder

Since users do not provide the vocal ability directly, we need to analyze their history singing records. Although the singing records could tell the best pitches they have sung, they could not directly determine the users' vocal range. To extract users singing information from their past singing, we proposed the history encoder to learn the vector embeddings from users' history records. As part of the key prediction model, the history encoder aims to learn the user embedding using an online fashion so that we could use it for key prediction in the next step.

In the history singing records $\mathbf{H}^i$ of each user, there are the sang notes $\mathbf{r}_t$. As described in Section 4.1, we use an embedding layer Φ to compute the vector representations $\mathbf{E}$ of the processed notes from the sang notes $\mathbf{r}_t$. As the notes in the song are time-dependent, the previous note affects the next note. We use the recurrent cells to analyze the user's history singing records so that we could learn a latent vector for their singing capability. Thus we propose to input $\mathbf{E}$ to recurrent cells such as gated recurrent units and use the output hidden vector as the latent representation for $\mathbf{E}$. Since users usually have more than one historical record, we take the mean of the hidden vector, and the result vector is denoted as $\mathbf{h}$.

4.3 Key Decoder

After computing the history vector from users' history singing records, we propose the key decoder to predict the target key. Besides the history vector, we need to input the target song, as we could analyze the song's notes sequences and recommend the best-fitted key to the user. The key decoder should consider both the user's singing ability and the characteristics of the songs.

Again, the key decoder aims to predict the probabilities of each key given the user's history vector $\mathbf{h}$ and the target song's notes $\mathbf{n}_{\text{s}}$. The transformer [40] has been a well-known language model capable of analyzing word sequences and has been successfully applied in many natural language understanding domains. Here we use the transformer encoder, denoted as TE, to combine the user's history information with the song information. Specifically, we first obtain the vector representation $\mathbf{E}_j$ of the target song using the same embedding layer Φ described in Section 4.1. Then we concatenate $\mathbf{h}$ at each note's embedding in $\mathbf{E}_s$ of the target song as $\mathbf{Z}_t = \{(\mathbf{e}_0, \mathbf{p}), (\mathbf{e}_1, \mathbf{p}), ..., (\mathbf{e}_l, \mathbf{p})\}$, and the input the resulting matrix $\mathbf{Z}_s$ to TE.

Next, we split the network into three separate decoders with independent weights for different losses. Each decoder is a dense layer and followed by different loss functions as described in Section 5. In detail, the value from the dense layer to the BCE loss ranges between 1 and 0, and it has a dimension of 1. We then concatenate it to the input vector to the other two dense layers. During the model inference, to recommend the key, we first look at the binary label for range classification, and we will only look at the probabilities for labels in the range determined by the range label. Specifically, the range label is determined by which range the key is placed. Since we treat it as a binary label, the range label 1 means the key is from the upper range, and the range label 0 means the key is from the lower range.

4.4 TRCOR Loss Functions

We use two losses to optimize the key prediction tasks for keys in 0 to -7 and keys in 0 to 7 respectively, denoted as L_1 and L_2. We also have a range classifier to determine if the target key is in the upper range (0 to 7) or lower range (0 to -7), where we use the binary cross-entropy loss L_3.

First, we extend the key labels into binary labels. Given a label k_i of the ith data, we convert it into two sets of binary labels. Since we have equal number of labels in the upper range and the lower range, we use m to denote the total number of labels in the each range. We will have two set of m binary labels, $\{k_i^{(0)}, ..., k_i^{(m-1)}\}$ for keys in the upper range and $\{k_i^{(0)}, ..., k_i^{(1-m)}\}$ for keys in the lower range, such that $k_i^{(m)} \in \{0, 1\}$ indicates whether k_i exceeds rank m. For example, if the key k_i is 3, we have a binary labels $k_i^{(3)} = \{1, 1, 1, 1, 0, 0, 0, 0\}$, where t means target label and 3 means the actual label 3, to input into the L_1 for keys in the upper range, and we have another binary labels {1, 0, 0, 0, 0, 0, 0, 0} to input into L_2 for keys in the lower range. Here $m = 8$ because we have a fixed number of keys in both ranges. Since the default key is 0 and we assume the key should be capable to sing with most people, we set it to be 1 in both binary labels no matter whether the key is from the upper range or not. That is, the default key 0 should always has the greatest probability. Thus, the model could give us a consistent non-decreasing predicted probability for both sets of binary labels. In other words, the keys from 0 to -7 will have a non-decreasing predicted probability, and the keys from 0 to 7 will also have similar monotonic probabilities.

In details, let $\mathbf{W_1}, \mathbf{W_2}$, and $\mathbf{W_3}$ denote the weight parameters of our model for each three tasks, and $\mathbf{b_1}$ and $\mathbf{b_2}$ denote the independent bias units that are added to the final output model to achieve the non-decreasing rank monotonicity. For the keys from the upper range (0 to 7), we have the loss function

$$\begin{aligned} L_1(\mathbf{W_1}, \mathbf{b_1}) = \\ -\sum_{i=1}^{N}\sum_{j=1}^{m-1}\big[\log(\alpha(g(\mathbf{x}_i, \mathbf{W_1}) + b_j))k_i^{(j)} \\ + \log(1 - \alpha(g(\mathbf{x}_i, \mathbf{W_1}) + b_j))(1 - k_i^{(j)})\big], \end{aligned} \tag{1}$$

where $g(\mathbf{x}_i, \mathbf{W}) + b_j$ is the output of penultimate layer added with the independent bias units b, α is the logistic sigmoid function, and the importance factor set to be 1. For the predicted probability of each binary label, we have $\alpha(g(\mathbf{x}_i, \mathbf{W_1}) + b_j)$ for each key in the range.

Similarly, for the keys from the lower range (0 to -7), we have the loss function

$$\begin{aligned} L_2(\mathbf{W_2}, \mathbf{b_2}) = \\ -\sum_{i=1}^{N}\sum_{j=1}^{m-1}\big[\log(\alpha(g(\mathbf{x}_i, \mathbf{W_2}) + b_j))k_i^{(1-j)} \\ + \log(1 - \alpha(g(\mathbf{x}_i, \mathbf{W_2}) + b_j))(1 - k_i^{(1-j)})\big], \end{aligned} \tag{2}$$

Then for the binary cross entropy loss used for range classification, we have

$$\begin{aligned} L_3(\mathbf{W_3}) = \\ -\sum_{i=1}^{N}\log(\alpha(f(\mathbf{x}_i, \mathbf{W_3})))r_i \\ + \log(1 - \alpha(f(\mathbf{x}_i, \mathbf{W_3})))(1 - r_i), \end{aligned} \tag{3}$$

where r_i is the range label for ith data and f is the linear layer. Then we adapt the way to compute the weighted sum of multiple loss values from [19] as the follows

$$\begin{aligned} L_{joint}(\mathbf{W_1}, \mathbf{W_2}, \mathbf{W_3}, \mathbf{b_1}, \mathbf{b_2}, \sigma_1, \sigma_2, \sigma_3) \approx \\ \frac{1}{2\sigma_1^2}L_1(\mathbf{W_1}, \mathbf{b_1}) + \frac{1}{2\sigma_2^2}L_2(\mathbf{W_2}, \mathbf{b_2}) + \frac{1}{\sigma_3^2}L_3(\mathbf{W_3}) \\ + \log\sigma_1 + \log\sigma_2 + \log\sigma_3, \end{aligned} \tag{4}$$

where σ is the variance which could be seen as the weight of each loss values. The large scale values of σ will decrease the contribution of loss, whereas the small scale value will increase the contribution. We initialize the log variance with zeros and train our model to predict the log variance because it is more numerically stable than regressing the variance.

By minimizing the joint loss L_{joint}, the independent bias units b are always non-increasing such that:

$$b_1 \geq b_2 \geq ... \geq b_{m-1} \tag{5}$$

for each $\mathbf{b_1}$ and $\mathbf{b_2}$ [6]. Then the predicted probability of each binary labels from each key range are non-increasing. Since $L_1(\mathbf{W_1}, \mathbf{b_1})$ is for the keys from the upper range (0 to 7), and $L_2(\mathbf{W_2}, \mathbf{b_2})$ is for the keys from the lower range (0 to -7), the predicted probability will increase from -7 to 0 and then decrease from 0 to 7, which guarantees the two-sided rank monotonicity.

4.5 Theoretical Guarantee

To prove the rank consistency in L_1 and L_2, we assume $(\mathbf{W}, \mathbf{b})$ is optimal, and we have $b_v < b_{v+1}$ for certain v where v is a rank. We claim that replacing b_v with b_{v+1} or vice versa would decrease the objective value L. First, we define the following variables:

$$\begin{aligned} A_1 &= \{i : k_i^{(j)} = k_i^{(j+1)} = 1\}, \\ A_2 &= \{i : k_i^{(j)} = k_i^{(j+1)} = 0\}, \\ A_3 &= \{i : k_i^{(j)} = 1, k_i^{(j+1)} = 0\}, \end{aligned}$$

where $A_1 \cup A_2 \cup A_3$ represents all possible situations for k in each the keys from upper range and lower range. Then we denote $p_i(b_j) = \alpha(g(x_i, \mathbf{W}) + b_j)$ and

$$\begin{aligned} \delta_i &= \log(p_i(b_{j+1})) - \log(p_i(b_j)), \\ \delta'_i &= \log(1 - p_i(b_j)) - \log(1 - p_i(b_{j+1})). \end{aligned}$$

Since L_3 does not have the bias term, the rank consistency guarantee is not related to it. In loss function Eq. (4), if we replace b_j with

Model	History Encoder	MAE	RMSE	ACC@1	ACC@3	P@1	P@3	R@1	R@3
Random	-	4.1805	5.1003	0.0726	0.2079	0.0678	0.2602	0.0675	0.1990
Ordinal Regression	-	2.4160	3.1565	0.1501	0.4067	0.1027	0.3802	0.0919	0.2621
Two Models	Dense	3.5026	4.5369	0.1107	0.2834	0.0449	0.2226	0.0837	0.2333
Two Models	BiGRU (Mean)	3.1556	4.1225	0.1265	0.3192	0.0929	0.2627	0.0853	0.2296
Ordinal Model	Dense	1.5764	2.5021	0.3018	0.6329	0.2170	0.5990	0.2134	0.4989
Ordinal Model	BiGRU (Dense)	2.1095	3.0137	0.2360	0.4930	0.1744	0.4865	0.1522	0.3612
Ordinal Model	BiGRU (Mean)	2.2908	3.2074	0.2061	0.4595	0.1501	0.3959	0.1526	0.3796
Ordinal Model	GRU	2.1989	2.9955	0.2120	0.4576	0.1818	0.4002	0.1327	0.3033
TRCOR	Dense	1.7484	2.6500	0.2377	0.5910	0.1801	0.5458	0.1614	0.4230
TRCOR	BiGRU (Dense)	2.3840	3.3133	0.1779	0.4455	0.1189	0.4021	0.1302	0.3297
TRCOR	BiGRU (Mean)	2.4408	3.3985	0.1725	0.4417	0.1232	0.4101	0.1370	0.3432
TRCOR	GRU	2.1978	2.9976	0.1832	0.4596	0.1141	0.4092	0.1105	0.2962

Table 1: Evaluation Results. The results indicate that under the two-sided rank-monotonicity constraint, our model has a better performance than the Two Models approach. Our model also has a very competitive performance against the Ordinal Model; whereas the Ordinal Model could not guarantee the two-sided monotonicity constraint.

b_{j+1}, the change of loss is denoted as $z()$ and is given as

$$\begin{aligned} z_1(L_{joint}) &= z_1(L_1) + z_1(L_2) \\ &= \Big[-\sum_{i\in A_1}\delta_i + \sum_{i\in A_2}\delta'_i - \sum_{i\in A_3}\delta_i \Big] \\ &\quad + \Big[-\sum_{i\in A_1}\delta_i + \sum_{i\in A_2}\delta'_i - \sum_{i\in A_3}\delta_i \Big]. \end{aligned}$$

Then if we replace b_{j+1} by b_j, the change of loss is given as

$$\begin{aligned} z_2(L_{joint}) &= z_2(L_1) + z_2(L_2) \\ &= \Big[\sum_{i\in A_1}\delta_i - \sum_{i\in A_2}\delta'_i - \sum_{i\in A_3}\delta_i \Big] \\ &\quad + \Big[\sum_{i\in A_1}\delta_i - \sum_{i\in A_2}\delta'_i - \sum_{i\in A_3}\delta_i \Big]. \end{aligned}$$

Now if we add $z_1(L_{joint})$ and $z_2(L_{joint})$, we have

$$z_1(L_{joint}) + z_2(L_{joint}) = -2\sum_{n\in A_3}(\delta_n + \delta'_n).$$

Since the change of b does not affect the loss function Eq. (3), and the right hand side of the equation is less than 0, we conclude that the claim is justified. Thus, the optimal solution would guarantee the decreasing rank monotonicity for the predicted probabilities for each of the key ranges in 0 to 7 and 0 to -7.

5 EXPERIMENTS

In this section, we will describe the experimental setups. In the dataset, users are having a different number of history records, so to maximize the training efficiency and make use of the batch processing, we set the length of history T to be the constant number 5. Then, we remove the users with less than 5 history records, and for other users, we randomly sampled T records to be the input to the model. For the training and testing data split, we use 70% of the dataset for training and 30% for testing.

5.1 Baseline Methods

First, we investigate the performance of the proposed TRCOR model against several baseline models. Due to the constraint specified in the previous sections, few previous works could be found, so we develop several baseline models for comparison. The details of the baseline models are as the following:

- **Random**: Among the 15 keys, we randomly select keys for the target users and songs.
- **Ordinal Regression**: We build a simple ordinal regression model based on the implementation described in [27]. To have a strong baseline model with a simple structure, we simplify the input data. Instead of using the prepossessing method described in Section 4.1, we concatenate the note number sequences in the history records with the note number sequences of the target song without using any Word2Vec model and encoding duration information. Then we input the concatenated note number vectors into the ordinal regression model and evaluate the model on key prediction. Note that the model does not perfectly fit our constraint as it could not guarantee the desired two-sided monotonicity.
- **Ordinal Model**: We also implement several other baseline models based on the OR-CNN [26]. Here we use the same model architecture described in Section 4, but we replace the CORAL loss and multitask learning part with the loss function from OR-CNN. Again, this model also could not guarantee the desired two-sided monotonicity.
- **Two Models**: To present a fair comparison, we develop a baseline model that could guarantee the desired rank-monotonicity, named the Two-Model approach. We first train two independent models using the CORAL loss from [6]. For both models, we use the same key recommendation model structure as described in Section 4. For one model, we train it using the data with only keys 0 to 7, and for another model, we train it using the data with only keys 0 to -7. Under this setting, each model could guarantee the rank-monotonicity such that there are decreasing predicted probabilities for keys from 0 to 7 and 0 to -7. When we evaluate this model, we have each model predict a key using the same input data,

Model	Hist Encoder	Average Number of Inconsistencies
Ordinal Regression	-	5.017
Two Models	Dense	0
Two Models	BiGRU (Mean)	0
Ordinal Model	Dense	4.238
Ordinal Model	BiGRU (Dense)	4.066
Ordinal Model	BiGRU (Mean)	4.518
Ordinal Model	GRU	4.105
TRCOR	Dense	0
TRCOR	BiGRU (Dense)	0
TRCOR	BiGRU (Mean)	0
TRCOR	GRU	0

Table 2: Evaluation results for inconsistencies occurred in the predictions. We compute the average numbers of rank inconsistencies of the models' output. We compare the numbers among all baseline models and TRCOR model, and we could conclude that TRCOR model could guarantee the two-sided rank consistency.

and we pick the key with a higher probability to be the final prediction.

5.2 Settings

For the history encoder in our key recommendation model, we implement four different encoders and compare the performance against each other. Note that the key decoders are the same for other models such as the Ordinal Model and Two Models for our baseline models.

- Dense: The Dense history encoder has two dense layers. After taking the mean of note embedding for the target song, we input the resulting vector to the dense layers. The first dense layer has a dimension of $(d, 64)$ where d is the dimension of the embedding layer, and it is followed by the Relu activation function. The second dense layer has a dimension of (64, 32), where 32 is the dimension of the history vector $\mathbf{h}$.
- GRU: The GRU history encoder has a GRU [14] layer with the input vector size to be d and the hidden vector size to be 32 which is the same as the dimension of the history vector. The output hidden vector from GRU is used to be the history vector $\mathbf{h}$
- BiGRU (Mean): The BiGRU (Mean) history encoder has a BiGRU [34] layer. The input vector size is d and the hidden vector size is 32. Since the BiGRU layer is bidirectional in terms of computing the hidden vectors, there are two hidden vectors. Here we use the mean of two hidden vectors to be the history vector $\mathbf{h}$.
- BiGRU (Dense): The BiGRU (Dense) history encoder also has a BiGRU layer, and it has the same setting as the BiGRU (Mean) for the input vector and hidden vector size. Here we handle the two hidden vectors output from the BiGRU layer in a different way. We concatenate two hidden vectors together, and input the resulting vector to a dense layer, followed by the Relu activation function. The output size of the dense layer is 32, and the vector output from the dense layer is seen as the history vector $\mathbf{h}$.

For all neural network-based models mentioned above, we optimize the loss functions with the Adam optimizer [20]. The other parameters include: batch size = 256, learning rate = 0.001, epochs = 20. For the transformer encoder, we set the dimension of the feed-forward network to be 200, the number of layers to be 2, the number of heads to be 2, and the dropout value to be 0.2. For evaluation metrics, we use the mean absolute error (MAE), root mean square error (RMSE), accuracy (ACC), accuracy@3 (ACC@3), precision (P), precision@3 (P@3), recall (R), and recall@3 (R@3).

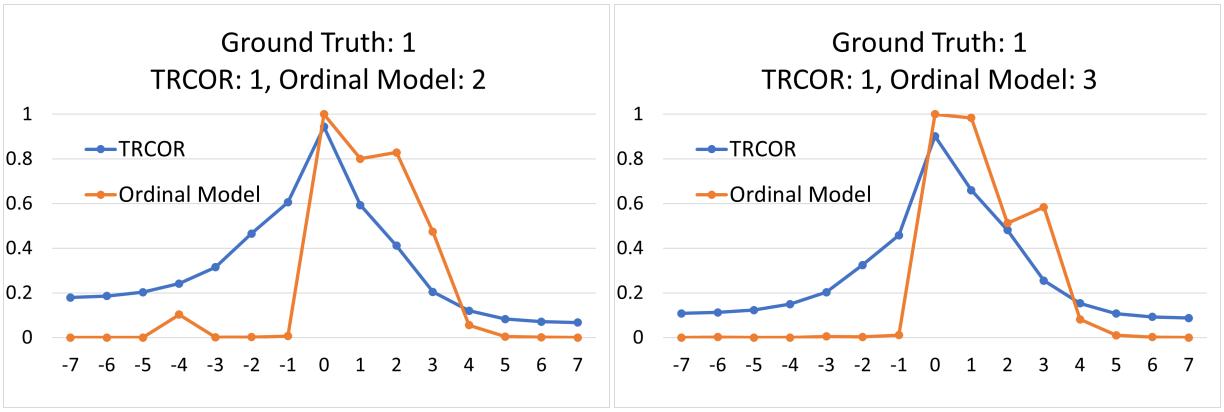
Due to the ordering information among the keys, the ACC@3, P@3, and R@3 are computed by comparing the ground truth key with the predicted key, predicted key + 1, and predicted key - 1.

5.3 Results and Analysis

5.3.1 Baseline Comparison. Table 1 presents the results of the baseline models against our proposed model. The results indicate that under the two-sided rank-monotonicity constraint, our model has a better performance than the Two Model approach. Our model also has a very competitive performance against the ordinal model; whereas the ordinal model could not guarantee the two-sided monotonicity constraint. To demonstrate the effectiveness of our proposed history encoder and key decoder, we compare the performance among the Ordinal Regression, the Ordinal Model, and TRCOR. For all variants of the history encoder, the neural network-based models have the better performance, which indicates that the history encoder could successfully extract the embedding representation from users' singing history and that the key decoder could accurately predict the key based on the history vector and the song embedding. Our model could successfully analyze the user's singing record the predict the best key for him or her to sing.

In Figure 3, we did a qualitative study on the output probabilities for keys. As shown in the graph, the TRCOR model successfully guarantees the two-sided monotonicity as desired such that the probabilities for keys from -7 to 0 are increasing and the probabilities for keys from 0 to 7 are decreasing. In contrast, the Ordinal model could not produce the desired probabilities for keys.

5.3.2 History Encoder. We implement four different history encoders which include Dense, GRU, BiGRU (Mean), and BiGRU (Dense). We expect the history encoder with recurrent cells to have a better performance, as the BiGRU and GRU should capture



(a) (b)

Figure 3: Examples of the predicted probabilities for each binary classifier tasks. Comparing between the TRCOR and Ordinal model, we can observe that the predicted probabilities of the keys from the lower range and upper range are increasing and decreasing for TRCOR model, whereas the predicted probabilities of Ordinal model is inconsistent.

Embedding Size	MAE	RMSE	ACC@1	ACC@3	P@1	P@3	R@1	R@3
64	1.7153	2.6067	0.2473	0.5979	0.1770	0.5369	0.1638	0.4257
128	1.7290	2.5841	0.2410	0.5912	0.1700	0.5366	0.1498	0.4066
256	1.7484	2.6500	0.2377	0.5910	0.1801	0.5458	0.1614	0.4230
512	2.3568	3.2078	0.1672	0.4270	0.1159	0.3650	0.1167	0.3162

Table 3: Evaluation Results for different embedding size. The results are obtained on TRCOR (Dense) model with different embedding size for the embedding layers. Among different metrics, the embedding size of 256 shows the best results, and we use embedding size of 256 for other models' experiments.

the time dependency information among the notes in the sequence. However, as the results in Table 1 show, the Dense history encoder outperforms other encoders and has significantly better performance in both Ordinal Model and TRCOR. Thus, we could claim that the time dependency within the sequence of the notes is not strong. A reason could be that the previous sang notes would not affect the future notes that the user would sing, as the competence of the note would majorly depend on the pitch range and singing ability of the user. Thus, using the simple Dense encoder is effective.

5.3.3 Two-sided rank consistency analysis. As described in Section 4, the TRCOR model guarantees the two-sided rank consistency. In this section, we analyze the rank inconsistency for the Ordinal Model. In Figure 3, we show four examples of the predicted probabilities output from TRCOR and the Ordinal Model. In Table 2, we summarize the average number of rank inconsistencies for the TRCOR, Ordinal Model, and Two Model. Since TRCOR and Two models both implemented the CORAL losses, the rank inconsistencies are 0 as expected. The Ordinal Model which implemented the loss function from [26] has positive values for the average number of inconsistencies which is a piece of evidence that the desired rank consistency could not be guaranteed. Thus, the observations show that the output from the TRCOR model is reasonable as the output probabilities follow the common sense that they are ordered as the keys.

5.3.4 Embedding Size. In Table 3, we evaluated the performance of the TRCOR (Dense) model with different embedding sizes including 64, 128, 256, and 512. We modify the embedding size for the embedding layer which is used to learn the vector representation for the notes. Among different embedding sizes, 256 seems to yield the best results overall. For all the models with the embedding layer, we do the experiment using the embedding size of 256.

6 CONCLUSION AND FUTURE WORK

In this paper, we develop a novel interpretable ordinal regression model for music key prediction. The proposed TRCOR model can guarantee the two-sided rank-monotonicity of the predicted probabilities over the keys, which makes the results explainable. The model can successfully extract the user singing capability from the historical singing records and combine it with the target song information. The experiments on the real-world karaoke dataset demonstrate the effectiveness of the proposed approach. In future work, we will apply the proposed two-sided rank consistent framework to other tasks that may demand two-sided rank monotonicity. For music key recommendation, we will explore transformers to extract history vectors from users' singing records and also study different key decoders and history encoders to further improve key recommendation.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering* 17, 6 (2005), 734–749.

[2] Oshin Agarwal, Yinfei Yang, Byron C Wallace, and Ani Nenkova. 2021. Interpretability analysis for named entity recognition to understand system predictions and how they can improve. *Computational Linguistics* 47, 1 (2021), 117–140.

[3] David Alvarez-Melis and Tommi Jaakkola. 2017. A causal framework for explaining the predictions of black-box sequence-to-sequence models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 412–421.

[4] Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction* 12, 4 (2002), 331–370.

[5] Paul-Christian Bürkner and Matti Vuorre. 2019. Ordinal regression models in psychology: A tutorial. *Advances in Methods and Practices in Psychological Science* 2, 1 (2019), 77–101.

[6] Wenzhi Cao, Vahid Mirjalili, and Sebastian Raschka. 2020. Rank consistent ordinal regression for neural networks with application to age estimation. *Pattern Recognition Letters* 140 (2020), 325–331.

[7] Wei Chu and S. Sathiya Keerthi. 2005. New Approaches to Support Vector Ordinal Regression. In *Proceedings of the 22nd International Conference on Machine Learning*. Association for Computing Machinery, New York, NY, USA, 145–152.

[8] Koby Crammer and Yoram Singer. 2002. Pranking with Ranking. In *Advances in Neural Information Processing Systems*, T. Dietterich, S. Becker, and Z. Ghahramani (Eds.), Vol. 14. MIT Press, 641–647.

[9] Piotr Dabkowski and Yarin Gal. 2017. Real time image saliency for black box classifiers. *Advances in neural information processing systems* 30 (2017).

[10] J Stephen Downie. 2003. Music information retrieval. *Annual review of information science and technology* 37, 1 (2003), 295–340.

[11] Mohamed Farah. 2009. Ordinal Regression Based Model for Personalized Information Retrieval. In *Conference on the Theory of Information Retrieval*. Springer, 66–78.

[12] Zeon Trevor Fernando, Jaspreet Singh, and Avishek Anand. 2019. A study on the Interpretability of Neural Retrieval Models using DeepSHAP. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1005–1008.

[13] Ferdos Fessahaye, Luis Perez, Tiffany Zhan, Raymond Zhang, Calais Fossier, Robyn Markarian, Carter Chiu, Justin Zhan, Laxmi Gewali, and Paul Oh. 2019. T-recsys: A novel music recommendation system using deep learning. In *IEEE International Conference on Consumer Electronics*. IEEE, 1–6.

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[15] Chu Guan, Yanjie Fu, Xinjiang Lu, Enhong Chen, Xiaolin Li, and Hui Xiong. 2017. Efficient karaoke song recommendation via multiple kernel learning approximation. *Neurocomputing* 254 (2017), 22–32.

[16] Chu Guan, Yanjie Fu, Xinjiang Lu, Hui Xiong, Enhong Chen, and Yingling Liu. 2016. Vocal Competence Based Karaoke Recommendation: A Maximum-Margin Joint Model. In *Proceedings of the SIAM International Conference on Data Mining*. 135–143.

[17] Ming He, Hao Guo, Guangyi Lv, Le Wu, Yong Ge, Enhong Chen, and Haiping Ma. 2020. Leveraging proficiency and preference for online Karaoke recommendation. *Frontiers of Computer Science* 14, 2 (2020), 273–290.

[18] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 1999. Support vector learning for ordinal regression. (1999).

[19] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7482–7491.

[20] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations*, Yoshua Bengio and Yann LeCun (Eds.).

[21] Yehuda Koren and Joe Sill. 2011. OrdRec: An Ordinal Model for Predicting Personalized Item Rating Distributions. In *Proceedings of the Fifth ACM Conference on Recommender Systems*. Association for Computing Machinery, New York, NY, USA, 117–124.

[22] G. Levi and T. Hassncer. 2015. Age and gender classification using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 34–42.

[23] Ling Li and Hsuan-tien Lin. 2007. Ordinal Regression by Extended Binary Classification. In *Advances in Neural Information Processing Systems*, B. Schölkopf, J. Platt, and T. Hoffman (Eds.), Vol. 19. MIT Press, 865–872.

[24] K. Mao, L. Shou, J. Fan, G. Chen, and M. S. Kankanhalli. 2015. Competence-Based Song Recommendation: Matching Songs to One's Singing Skill. *IEEE Transactions on Multimedia* 17, 3 (2015), 396–408.

[25] Tomas Mikolov, Kai Chen, G. Corrado, and J. Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations*.

[26] Zhenxing Niu, Mo Zhou, Le Wang, Xinbo Gao, and Gang Hua. 2016. Ordinal Regression With Multiple Output CNN for Age Estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[27] Fabian Pedregosa-Izquierdo. 2015. *Feature extraction and supervised learning on fMRI : from practice to theory*. Theses. Université Pierre et Marie Curie - Paris VI.

[28] Colin Raffel. 2016. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Ph. D. Dissertation. Columbia University.

[29] Shyamsundar Rajaram, Ashutosh Garg, Xiang Sean Zhou, and Thomas S. Huang. 2003. Classification Approach towards Ranking and Sorting Problems. In *Machine Learning: European Conference on Machine Learning*, Nada Lavrač, Dragan Gamberger, Hendrik Blockeel, and Ljupčo Todorovski (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 301–312.

[30] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

[31] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, Arlington, Virginia, USA, 452–461.

[32] R. Rothe, R. Timofte, and L. Van Gool. 2015. DEX: Deep EXpectation of Apparent Age from a Single Image. In *IEEE International Conference on Computer Vision Workshop*. 252–257.

[33] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web*. Springer, 291–324.

[34] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing* 45, 11 (1997), 2673–2681.

[35] Amnon Shashua and Anat Levin. 2002. Ranking with Large Margin Principle: Two Approaches. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, 961–968.

[36] Libin Shen and Aravind K Joshi. 2005. Ranking and reranking with perceptron. *Machine Learning* 60, 1-3 (2005), 73–96.

[37] Jaspreet Singh and Avishek Anand. 2020. Model agnostic interpretability of rankers via intent modelling. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 618–628.

[38] Bing-Yu Sun, Jiuyong Li, Desheng Dash Wu, Xiao-Ming Zhang, and Wen-Bo Li. 2009. Kernel discriminant learning for ordinal regression. *IEEE Transactions on Knowledge and Data Engineering* 22, 6 (2009), 906–910.

[39] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, Vol. 26. Neural Information Processing Systems Foundation, 9.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc.

[41] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. 2006. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. 501–508.

[42] Xinxi Wang and Ye Wang. 2014. Improving Content-Based and Hybrid Music Recommendation Using Deep Learning. In *Proceedings of the 22nd ACM International Conference on Multimedia*. Association for Computing Machinery, New York, NY, USA, 627–636.

[43] Yuan Wang, Shigeki Tanaka, Keita Yokoyama, Hsin-Tai Wu, and Yi Fang. 2021. Karaoke Key Recommendation Via Personalized Competence-Based Rating Prediction. In *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 286–290.

[44] Yongfeng Zhang, Xu Chen, et al. 2020. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval* 14, 1 (2020), 1–101.