ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-7605-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICASSP39728.2021.9414524

KARAOKE KEY RECOMMENDATION VIA PERSONALIZED COMPETENCE-BASED RATING PREDICTION

Yuan Wang[1] *Shigeki Tanaka*[2] *Keita Yokoyama*[3] *Hsin-Tai Wu*[3] *Yi Fang*[1]

[1]Santa Clara University, Santa Clara, CA, USA
[2]NTT DOCOMO, INC., Tokyo, Japan
[3]DOCOMO Innovations, Inc., Palo Alto, CA, USA

ABSTRACT

Karaoke machines have become a popular choice for many people's daily entertainment. In this paper, we address a novel task of recommending a suitable key for a user to sing a given song to meet his or her vocal competence, by proposing the Personalized Competence-based Rating Prediction (PCRP) model. Specifically, we learn the song embedding vectors from the sequences of songs' notes, and then design a history encoder with recurrent units to extract users' vocal information from the history rating records and utilize a rating decoder based on the Transformer. The experimental results on a real world karaoke rating dataset demonstrate the effectiveness of the proposed approach.

Index Terms— Representation learning, Encoder decoder architecture, Music recommendation systems

1. INTRODUCTION

Karaoke has become an important form of entertainment found worldwide. To deliver a desired performance for a target song, a user had better select a suitable key so that the song could meet his or her vocal competence. In music, a key is the scale around which a piece of music revolves, and each key has seven notes which make up the song's melody. There are a total of 15 keys ranging from 0 to 14, with the default key to be 7. The original key of each song is set to fit the professional singer's vocal competence, and it is usually difficult for non-professionals to sing with that key. For most users who do no have musical knowledge, they do not know how to choose a suitable key. Thus, to help each individual user decide the best key for the target song becomes an important task. There exist some works [1, 2, 3, 4] on karaoke song recommendation which aims to recommend the songs that would fit the users' vocal competence instead of their taste preferences. However, all these works focus on recommending songs, and ignore the fact that the key of a song is also an important factor during the recommendation.

In collaboration with Daiichikosho, one of the largest producers of karaoke machinery in the world, we are provided users' historical rating records and the sequences of songs' notes in the automatic rating system. Note that the data are processed in MIDI format which is commonly used in music generation. For users, only one final rating is presented, which is calculated based on the completion and perfection of the singing record. Internally, a more detailed evaluation is given by the rating system in the karaoke machine. Each song is segmented into a fixed number of sections, and each section receives a rating which ranges between 0 and 100. Each section of the song contains multiple notes, and the length of the section is predefined by the system. Usually, when the user sings on a karaoke machine, their singing are recorded for analysis. Since the users' singing records are often not logged by the machine, the users' vocal characteristic can be analyzed and extracted from their history rating records.

Inspired by the success of representation learning [5, 6, 7, 8] for natural language understanding and computer vision, we propose an efficient and semantic representation learning method for the notes in songs. Recently some music representation learning models are proposed to seek to learn semantic vector representation for music based on convolutional neural network (CNN) [9, 10, 11, 12], Transformer [13], WaveNet [14], and VAE [15]. However, these models rely on the music's spectrograms or MIDI files, which is normally not available for the vocal only recordings in our problem setting.

To the best of our knowledge, this paper presents the first study of karaoke key recommendation. We propose the Note Section Embedding with Key (NSEK) for learning an efficient and semantic representation containing the key, note, and time interval information. To recommend a key given a specific user and a song, we propose the Personalized Competence-based Rating Prediction model (PCRP) which can analyze the user' history rating records and accurately predict the rating for the target key and song without the need for vocal recording from the user and song. The experimental results on a real world Karaoke dataset demonstrate the effectiveness of the proposed approach. The source code will be made public upon paper acceptance.

2. MUSIC REPRESENTATION LEARNING

2.1. Data Characteristics

The karaoke dataset provided by the company includes the users' history rating records $\mathbf{H}$ and the notes for songs' vocal track $\mathbf{N}$. There are a total of 11,433 users, 480,826 history

rating records, and 1,043 songs. The default key 7 has the most cases, which again indicates the problem that most users do not know how to change keys to achieve the best performance.

For each j^{th} song in $\mathbf{N}$, $\mathbf{n}_j$ is a sequence of tuples where each tuple contains the note number, duration, and section number. Specifically, the sequence of notes contains only the notes for the vocal track of the songs. For each t^{th} record in $\mathbf{H}^u$ for a user u, there are the target song $\mathbf{n}_t$, the target key k_t, the ratings $\mathbf{r}_t$, the singing technique summary $\mathbf{a}_t$, so that $\mathbf{H}_t^u = (\mathbf{n}_t, k_t, \mathbf{r}_t, \mathbf{a}_t)$. Also, each user u had T history records where $T = |\mathbf{H}^u|$. $\mathbf{r}$ is a l-dimension vector since the karaoke machine gives one rating for each of the l sections. The singing technique summary $\mathbf{a}$ is given automatically by the karaoke machine's rating system, which has 5 columns of statistics of the all previous singing performances. For example, it includes the average of number of pitches that users have sung perfectly from the past records. The larger values in the statistics indicates the better performance.

2.2. Note Section Embedding with Key

In this section, we describe our proposed music notes representation learning model: Note Section Embedding with Key (NSEK). Since the original sequences of notes contain only the discrete value of notes, we need an informative representation for the songs so that the note, the duration, and the key could all be encoded together. Inspired by Word2Vec [5] originally proposed for words representation learning, our NSEK learns a low-dimensional representation containing the desired information for each song.

We first present a method to encode the duration information. We set a standard minimum time interval and compute the number of the interval in this duration by dividing the duration by the interval and rounding to the closest integer. Then we convert each note number to the text and duplicate the note text by that number to get a note word w. For example, given a tuple $\{65, 0.003, \text{Sec } 0\}$ where 65 is the note number and 0.003 is the duration, and the standard interval of 0.001, the note and the duration are jointly represented by '65_65_65' as repeating '65' by three times. Thus, for each sequence of notes $\mathbf{n}_j$, we can get a sequence of note words $\mathbf{w}_j = \{w_1, ..., w_{|\mathbf{w}_j|}\}$. We learn a low-dimensional representation for the note words to use the models from Word2Vec [5]. Specifically, each word in the sequence $\mathbf{w}_j = \{w_1, ..., w_{|\mathbf{w}_j|}\}$ is encoded as $\mathbf{g} = \Phi(w)$ where $\mathbf{g}$ is the d-dimensional embedding and d is the dimension of the embedding, encoding the note and the duration information together. After learning the embedding $\mathbf{g}$ for each word in $\mathbf{w}_j$, we then utilize the section information to compute the representation vector for each section in $\mathbf{n}_j$. To obtain the section embedding $\mathbf{E}_j$, we calculate the average of normalized notes embedding from the same section. Since the total number of sections for each song is same, $\mathbf{E}_j$ is a $l \times d$ matrix so that $\mathbf{E}_j = \{\mathbf{e_{j,0}}, \mathbf{e_{j,1}}, ..., \mathbf{e_{j,l}}\}$. In default, the note number is always under the key 7, and the new target key changes the note number by adding or subtracting the difference between the new key and the default key. For example, if we change the key to 8, we just need to add 1 to all note number. After getting the new sequence of notes with the key, we perform the same steps described above to obtain NSEK. With the target key k, NSEK becomes $\mathbf{E}_j(k)$ indicating the $\mathbf{E}_j$ under the key k. In this way, our proposed NSEK could integrate the key, duration, and time interval information together in the low-dimensional embedding vectors.

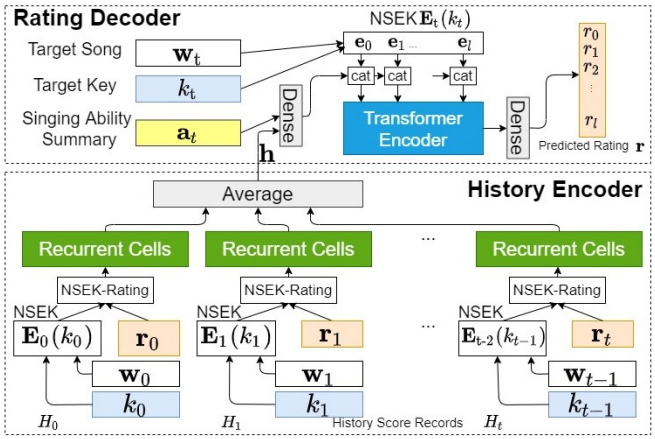

Fig. 1: Architecture of the PCRP model

It is worth noting that we use Word2Vec [5] to learn note embedding instead of more recent and advanced language models such as BERT [16] and GPT-2 [17]. The main reason is that our dataset contains only about one thousand songs, which is quite small compared to the text corpus used to train these deep language models.

3. PERSONALIZED COMPETENCE-BASED RATING PREDICTION MODEL

In this section, we propose the Personalized Competence-based Rating Prediction (PCRP) model for the key recommendation. Our setting assumes the user have had at least one record. We address the aforementioned problem using solely the history rating records and the NSEK learned from the sequences of songs' notes. The PCRP model has the encoder-decoder structure, which consists of a history encoder and a rating decoder. The history encoder aims to analyze the user's history records so that the rating decoder could predict the ratings on top of the encoder.

At each time T, the user u who has history rating records $\mathbf{H}^u$, selects a target song with n_{target}. The history encoder first computes a history vector $\mathbf{h}_T$ using the history rating records $\mathbf{H}^u = \{\mathbf{H}_0, \mathbf{H}_2, ...\mathbf{H}_{T-1}\}$. Then for each $k \in \mathbb{Z}^+ \leqslant 14$, we input the target NSEK $\mathbf{E}_t(k)$ of the song notes $\mathbf{n}_t$, the history vector h_T and the singing ability summary $\mathbf{a}_T$ to the rating decoder which will predict the rating $\hat{\mathbf{r}}$. Then the key with the highest predicted ratings will be recommended to the user u for the target song.

3.1. History Encoder

In the history rating records $\mathbf{H}^u$, each tuple contains the ratings $\mathbf{r}_t$, the chosen key k_t and the song $\mathbf{n}_t$. After computing the NSEK $\mathbf{E}_t(k)$ with the k from the song $\mathbf{n}_t$, how to combine the two vectors $\mathbf{E}_t(k)$ with $\mathbf{r}_t$ becomes a question. We want to extract the user's singing ability information from the ratings which are associated with the song notes. Thus, on the top of the NSEK, we propose NSEK-rating which further integrates the rating information in the embedding. Since the rating for each section ranges from 0 to 100, we could use ratings as the weight values that could be added to the embedding vector for each section. The user's performance ratings are automatically generated by the rating system, and intuitively the ratings could denote the level of completion of each section. Thus, we compute NSEK-rating $\mathbf{Q}_t$ as the weighted NSEK using the rating per section as $\mathbf{Q}_t = \mathbf{r}_t \odot \mathbf{E}_t(k)$ where $\odot$ is the element-wise product and each rating value in $\mathbf{r}_t$ are normalized into the range 0 to 1. Under this transformation, the NSEK-rating represents both the notes information and the performance information. Then we propose to input each NSEK-rating $\mathbf{Q}_t$ to recurrent cells, and use the output hidden vector as the latent representation for $\mathbf{Q}_t$. For users with more than one history rating records, we take an mean of the hidden vectors. Specifically, we use the recurrent unit with the share weights to output a hidden vector $\mathbf{v}$ for each $\mathbf{Q}_t$, and we compute the history vector $\mathbf{h}$ as $\mathbf{h} = \frac{1}{T}\sum_{i=0}^{T-1}\mathbf{v}_i$.

3.2. Rating Decoder

The rating decoder aims to use the history vector $\mathbf{h}$ to predict the rating for each section in the target song. The transformer [18] has already shown its capability of analyzing word sequences and been successfully applied in many domains such as question answering. Thus, we use the transformer encoder, denoted as TE, to encode the user's history information with the song information. Specifically, as illustrated in Figure 1, besides the history vector $\mathbf{h}$, we also make the use of the singing ability summary $\mathbf{a}_t$ by concatenating it with $\mathbf{h}$, and a dense layer $\mathcal{F}$ is applied to encode the concatenated vector such that $\mathbf{p} = \mathcal{F}(\mathbf{h}, \mathbf{a}_t)$. Since the resulting vector $\mathbf{p}$ is the low-level representation of the user's past performance summary, we then concatenate $\mathbf{p}$ at each section's embedding in $\mathbf{E}_j(k)$ 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 $\mathbf{Z}_j$ to TE. Here we only used the transformer encoder because we believe the time dependency between ratings in different sections is weak, so a dense layer is used to output the ratings. The performance in the previous section will not affect the ratings in the next section because the singing records are evaluated by the system automatically section by section. The result vector output from the transformer encoder is passed through the ReLu activation [19] followed by a dense layer $\mathcal{F}$ of size $d \times l$ where d is the dimension of the result vector. Thus, the output vector from the dense layer is the sequence of ratings for the target song. Specifically, the target predicted rating $\hat{r}$ is calculated by $\hat{\mathbf{r}} = \mathcal{F}(Relu(TE(\mathbf{Z}_t)))$.

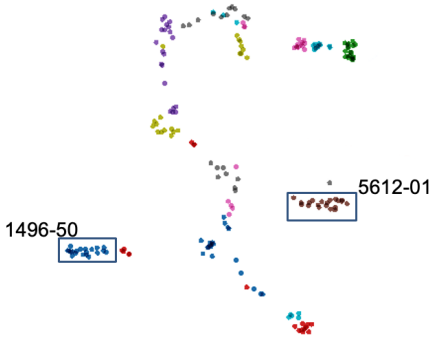

Fig. 2: Visualization of the song section embeddings. The embeddings from the same song are marked in the same color.

3.3. Recommendation

Since the final output from the model is the sequence of ratings $\hat{\mathbf{r}}$ for the target song and key, it does not directly indicate the best key for the target song. The output ratings from the model are used to show the predicted performance of the user under the target key. To perform recommendation, we need to predict ratings for each of 15 keys and rank the key by the ratings summed from all sections. The key with the best rating is presented to the user as $\hat{k} = \text{argmax}_k \sum_{i=0}^{l-1} \hat{r}_i$.

4. EXPERIMENTAL RESULTS

4.1. Note Embedding

Before we discuss the performance of PCRP, we first evaluate the note embedding with NSEK proposed in Section 2.2 to check if the learned embedding encodes desired information. To learn the note words embedding, we train with the Fasttext skip gram model [20], and conduct evaluation through the k-nearest-neighbors. The intuition is as follows, the note section embedding from the same song should be closer to each other in the embedding space. We randomly select 10 songs and plot the note section embedding vectors learned by NSEK in the 2-dimension space using UMAP [21] as shown in Figure 2. Note that the segments embedding from the same song are given the same color, and two example song IDs are marked. From the figure, it is clear that the section embeddings from the same song are clustered together, which indicates the effectiveness of NSEK.

4.2. Rating Prediction and Key Recommendation

For a fair comparison and having the same validation dataset for different length of history rating records, we split the training and validating dataset based on the length of users' history

Models		MSE	NDCG@3	NDCG@5	P@3	P@5	MRR
Linear Regression		185.504	0.082	0.132	0.119	0.238	0.174
Transformer Encoder		110.250	0.151	0.253	0.272	0.530	0.211
PCRP	Dense	84.456	0.183	0.301	0.285	**0.563**	0.259
	GRU	84.640	0.250	**0.310**	0.285	0.430	0.327
	BiGRU(Avg)	**82.628**	**0.274**	0.297	**0.318**	0.377	**0.350**
	BiGRU(Dense)	83.906	0.179	0.216	0.285	0.377	0.237

Table 1: Experimental results of rating prediction in MSE and of key recommendation in other metrics

records. The users who have more than 700 records are used for validation, which has a total of 68,343 records (15%) from the entire dataset. Below are the baseline methods for comparison in the experiments:

- Linear Regression: We flatten the NSEK embedding and concatenate it with $\mathbf{a}_t$ and then input the resulting vector to the linear regression model to predict the l dimension rating vector.
- Transformer Encoder: We use the transformer encoder [18] as the rating decoder. We pass only $\mathbf{a}_t$ through the dense layer such that $\mathbf{r} = \mathcal{F}(\mathbf{a}_t)$. Other components remain the same.

We also experiment with four variants of PCRP with different history encoders. During the training, we minimize the mean square error loss (MSE) function with the Adam optimizer [22]. The other parameters of the model include: batch size=256, learning rate = 0.001, epochs = 15.

- Dense: We concatenate the NSEK embedding and ratings and pass the concatenated vector through a dense layer. The resulting vector is used for history vector $\mathbf{h}$.
- GRU: We input the NSEK-rating to GRU [23] and use the output hidden vector as $\mathbf{h}$.
- BiGRU(Avg): We use the BiGRU [24] as the recurrent cell, and input the NSEK-rating to BiGRU. Then we use the average of the output hidden vector as $\mathbf{h}$.
- BiGRU(Dense): We also propose to flatten the output hidden vectors from the BiGRU and pass it through a dense layer to get the history vector $\mathbf{h}$.

To evaluate the predicted final rating of user's singing performance, we use the mean square error (MSE) [25] by comparing the predicted rating against the actual rating. The evaluation of the recommended keys is more tricky because we do not know the optimal key for the given song and user. However, we observe that only less than 1% cases have an average rating greater than 95 (out of 100). These high ratings which indicate the excellent performance delivered by the user can be viewed as the best keys for those cases. The proposed model predicts the ratings for all the keys (from 0 to 15) and ranks the keys in the descending order of the predicted ratings (averaging over all the sections in a song). We can then evaluate the ranked key list for each user of the test data using the standard ranking based evaluation metrics for recommendation systems such as normalized discounted cumulative gain (NDCG) and mean reciprocal rank (MRR) [26].

Table 1 contains the results of our proposed model against the baselines for rating prediction (in MSE) and recommendation (in NDCG and MRR). As we can see, PCRP substantially reduced the rating prediction error in MSE, compared with linear regression and Transformer. The improvements in terms of MSE show that analyzing the history rating records helps the model better understand the user's singing ability, thereby leading to a more accurate rating prediction. The history rating records contain the underlying user information which could be extracted by our proposed history encoder. As another reference point, if we use the average of each user's historical ratings for prediction, the resulting MSE is 539.633, which is much higher than the machine learning based methods.

Regarding the recommendation performance, we compare the top-k recommendation performance using the NDCG@k, Precision@k, and MRR. As shown in Table 1, all the variants of PCRP yield much better results than the baselines, which demonstrates the advantage of using history encoder in PCRP. Among the four variants, BiGRU(Ave) produces the best results in NDCG@3, P@3, and MRR, which indicates the usefulness of the Bidirectional GRU units with averaging the output hidden vectors. The overall results have a similar pattern as the rating evaluation results, which shows the effectiveness of PCRP in the recommendation task. This is also consistent with the intuition that better rating predictions will lead to better performance in recommendation.

5. CONCLUSION

This paper tackles a novel personalized recommendation task: karaoke key recommendation. We leverage the success of representation learning in natural language processing to encode the key, note, and time interval information in a song. Based on the learned music embeddings, an encoder-decoder architecture is proposed to make predictions on users' final performance on a given song and key. The experiments on a real world karaoke dataset demonstrates the effectiveness of the proposed approach.

6. REFERENCES

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

[2] Ming He, Hao Guo, Guangyi Lv, Le Wu, Yong Ge, Enhong Chen, and Haiping Ma, "Leveraging proficiency and preference for online karaoke recommendation," *Frontiers of Computer Science*, vol. 14, no. 2, pp. 273–290, 2020.

[3] Chu Guan, Yanjie Fu, Xinjiang Lu, Hui Xiong, Enhong Chen, and Yingling Liu, "Vocal competence based karaoke recommendation: A maximum-margin joint model," in *ICDM*, 2015, pp. 135–143.

[4] K. Mao, L. Shou, J. Fan, G. Chen, and M. S. Kankanhalli, "Competence-based song recommendation: Matching songs to one's singing skill," *IEEE Transactions on Multimedia*, vol. 17, no. 3, pp. 396–408, 2015.

[5] Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean, "Efficient estimation of word representations in vector space," *Proceedings of Workshop at ICLR*, vol. 2013, 01 2013.

[6] Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc V Le, "Semi-supervised sequence modeling with cross-view training," *EMNLP*, 2018.

[7] Khurram Javed and Martha White, "Meta-learning representations for continual learning," in *NeurIPS*, 2019, pp. 1820–1830.

[8] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine, "Near-optimal representation learning for hierarchical reinforcement learning," *ICLR*, 2018.

[9] John Thickstun, Zaïd Harchaoui, and Sham M. Kakade, "Learning features of music from scratch," in *ICLR*, 2017.

[10] Alexander Schindler and Peter Knees, "Multi-task music representation learning from multi-label embeddings," in *CBMI*. IEEE, 2019, pp. 1–6.

[11] Jiyoung Park, Jongpil Lee, Jangyeon Park, Jung-Woo Ha, and Juhan Nam, "Representation learning of music using artist labels," *ISMIR*, 2018.

[12] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," *ISMIR*, 2017.

[13] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, and Douglas Eck, "Music transformer: Generating music with long-term structure," *ICLR*, 2018.

[14] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[15] Gino Brunner, Andres Konrad, Yuyi Wang, and Roger Wattenhofer, "Midi-vae: Modeling dynamics and instrumentation of music with applications to style transfer," *ISMIR*, 2018.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *NAACL*, 2019.

[17] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, pp. 9, 2019.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *NeurIPS*, 2017, pp. 5998–6008.

[19] Vinod Nair and Geoffrey E Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[20] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov, "Enriching word vectors with subword information," *TACL*, 2017.

[21] Leland McInnes, John Healy, and James Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[22] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.

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

[24] Mike Schuster and Kuldip K Paliwal, "Bidirectional recurrent neural networks," *IEEE TSP*, vol. 45, no. 11, pp. 2673–2681, 1997.

[25] Tianfeng Chai and Roland R Draxler, "Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature," *Geoscientific model development*, vol. 7, no. 3, pp. 1247–1250, 2014.

[26] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press, USA, 2008.