TRUNCATED SVD-BASED FEATURE ENGINEERING FOR SHORT VIDEO UNDERSTANDING AND RECOMMENDATION

Tsun-Hsien Tang[12], *Kuan-Ta Chen*[2], *Hsin-Hsi Chen*[13]

[1]Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
[2]Institute of Information Science, Academia Sinica, Taipei, Taiwan
[3]MOST Joint Research Center for AI Technology and All Vista Healthcare, Taipei, Taiwan
thtang@nlg.csie.ntu.edu.tw; swc@iis.sinica.edu.tw; hhchen@ntu.edu.tw

ABSTRACT

Short video app, like TikTok[1], has received wide acclaim due to the prevalence of social media and the availability of recording devices such as mobile phones. Moreover, with the advent of the big data age, the use of historical user behaviors from multi-modal resources plays a pivotal role in the video recommendation system. In the ICME 2019 Short Video Understanding Challenge, participants are asked to predict whether a user will *finish* and *like* a specific short video along with its multi-modal features, i.e., the problem is formulated as a click-through rate prediction task. In this paper, we present an ensemble of unconventional models to the task, including tailored neural networks structure based on Compressed Interaction Network (CIN) and Gradient Boosting Decision Trees (GDBTs) using classic SVD-based features. We achieved a weighted AUC score of 0.8029 and 0.8037 on the Public and Private Leaderboard[2] of track2, respectively, and ended up with the 3^{rd} place in the competition.

Index Terms— Short Video Recommendation, Deep Learning, Gradient Boosting Decision Tree

1. INTRODUCTION

Powered by high speed network and affordable recording devices, short video streaming now becomes an engaging and emerging type of service in social media. People not only enjoy short videos, but also share a clip of life in a second on the social media such as TikTok and Instagram[3]. Large amount of user-generated content can thus be displayed on a tiny screen. However, the overload of information would heavily harm the user experiences.

Several successful social media platforms have shown the value of feeding relevant information to users. Therefore, to grab the users' attention, recommender systems are adopted to match user interests with resource items. Conventional recommendation methods, e.g., matrix factorization [1], mainly model users preference towards items using historical user-item interaction records.

Recently, various kinds of auxiliary data become increasingly available. In addition, inspired by the immense success of deep learning in computer vision [2] and natural language processing [3, 4], deep learning based methods [5, 6, 7] have been proposed for CTR (click-through rate) prediction task. As a result, to describe user portraits and item properties in more detail, multi-model features extracted using deep learning models from auxiliary data have a significant impact on enhancing prediction performance.

Given multi-modal video features, including visual features, text features and audio features, as well as user interactive behavior data, such as *finish* and *like*, the ICME 2019 Short Video Understanding Challenge cooperating with ByteDance Inc. asked the ML community to predict the user's click behavior (*finish* or *like*) on a collected video dataset, named Byte-Recommend100M. Two sub-tracks with various scales of datasets are proposed; track1 contains hundreds of millions of data and track2 is a relatively small-scale dataset with tens of millions of data. Moreover, a baseline implementation of factorization machine [8] using 5 features (`user_id, user_city, item_id, author_id, item_city`) is kindly released .[4]

In this paper, we present our solution to reach the 3^{rd} place in track2. Based on the fact that the probability of *finish* is highly correlated to that of *like*, we incorporate gradient boosting decision trees and deep neural networks, which are trained in a multi-task learning fashion, to make recommendations. Additionally, since feature engineering plays a crucial role in the success of many predictive systems, we further illustrate the techniques for feature creation. Extensive experimental results demonstrate the road map to achieve 0.8038 on Private Leaderboard.

[1]https://www.tiktok.com/
[2]https://biendata.com/competition/icmechallenge2019/final-leaderboard/
[3]https://www.instagram.com/
[4]https://github.com/challenge-ICME2019-Bytedance/Bytedance_ICME_challenge

978-1-5386-9214-1/19/$31.00 ©2019 IEEE
DOI 10.1109/ICMEW.2019.00136

2. RELATED WORK

Video recommender systems have triggered researchers' interests in recent years. Davidson et al. [9] made use of association rule mining [10] and co-visitation counts to compute the relatedness score of two videos. He et al. [11] develop techniques based on neural networks to tackle the key problem in recommendation - collaborative filtering - on the basis of implicit feedback.

With the help of plentiful meta-data, advanced deep learning based models are developed to use multi-modal features without much effort on feature engineering. For example, Covington et al. [12] proposed a deep candidate generation model to encode the arbitrary continuous and categorical features into dense representations, which is a memory-efficient way comparing to traditional matrix factorization methods. Not limited to video recommender system, a variety of click-through-rate problems could also be solved using deep learning neural networks. Guo et al. [5] proposed a model, DeepFM, to combine the power of factorization machines for recommendation and deep learning for feature learning in a new neural network architecture. Lian et al. [6] proposed a novel Compressed Interaction Network (CIN), which aims to generate feature interactions in an explicit fashion and at the vector-wise level. Wang et al. [7] proposed the Deep & Cross Network (DCN) equipped with a novel cross network that is more efficient in learning certain bounded-degree feature interaction.

In this paper, we introduce a tailored structure of neural networks to deal with the multi-task learning techniques. The proposed architecture can furnish a complementary prediction to GBDTs and boost the performance by ensembles.

3. APPROACH

3.1. Data description

Byte-Recommdend100M dataset, provided by ByteDance Inc., consists of tens of thousands of different users and 100 Millions of different videos. More specifically, in track2, 19,622,340 samples are used for training and another 2,761,799 samples for testing. The multi-modal features for each sample are list below:

- Face features including beauty score, gender and the position of face displayed in the video
- The video content is modeled by 128-dimensional visual features.
- The back ground music (BGM) is also represented by 128-dimensional audio features.
- The words of title are first encoded and transformed into format of bag-of-words representation.

For the interaction data between user and video, the specific meaning of each field is shown in Table 1. `finish` and `like` are treated as prediction target in this task.

Table 1: Description of interaction data

Field name	Field description	Data type
`uid`	User id	int
`user_city`	User's city	int
`item_id`	Video id	int
`author_id`	Author id	int
`item_city`	Video city	int
`channel`	Channl of the video	int
`finish`	finish the video or not	bool
`like`	like the video or not	bool
`music_id`	Music id	int
`did`	Device id	int
`creat_time`	Video release time	int
`video_duration`	Video duration	int

3.2. Feature engineering

Here we explain the non-trivial feature sets we use in order to describe the user portrait or item content comprehensively.

3.2.1. One-hot encoding

To begin with, we encode the categorical feature using a one-hot (aka one-of-K or dummy) encoding scheme. This operation consumes so much memory that we use compressed sparse row matrix for storage instead of the dense one. Binary vectors with 5,212,602 dimensions are then generated.

3.2.2. Latent features

The classical recommendation algorithms is based on the idea of matrix factorization, which is superior to nearest-neighbor techniques. To obtain the dense representations for users and items (videos, musics and authors here), we construct latent features from the user-item interaction matrix via singular value decomposition [13]. Given an interaction matrix $A \in R^{n \times m}$, where n and m denote the number of unique users and items, respectively, the SVD operation could be express as the product of three matrices,

$$A = USV^T \tag{1}$$

where $U \in R^{n \times n}$ and $V \in R^{m \times m}$ are two orthogonal matrices and represent the latent features for users and items, respectively; and $S \in R^{n \times m}$ is a diagonal matrix with non-negative entries known as the singular values.

Apart from (`uid`, `item_id`) interaction pairs, we further consider (`uid`, `music_id`) and (`uid`, `author_id`) to construct the matrix in our solution. The latent features are thus able to depict the user portraits through various aspects. In addition, the item could be well represented by latent factors extracted from V. Moreover, since the original U and V matrices are too large to fit in the memory, low-rank approximation technique is adopted to reduce the latent dimension for

feature construction; that is, we would like to build a rank-k approximation matrix A_k obtained from the truncated singular value decomposition (selecting the k most significant singular components),

$$A_k \quad = \quad U_k S_k V_k^T, \tag{2}$$

so that $A \approx A_k$. The dimensions of U and V^T become $n \times k$ and $m \times k$. The low rank approximation not only reveals hidden relationships between users and items (video, music and author) in an efficient way, but also reduces the risk of overfitting since only important information is retained. From a pragmatical point of view, truncated SVD is fast and the decomposition result is unique. That is a nice property for reproducing. Finally, latent features with diverse characteristics are generated using interaction between (`uid`, `item_id`) with $k = 50$, (`uid`, `music_id`) with $k = 10$ and (`uid`, `author_id`) with $k = 10$.

Another method to create latent feature is to find the embedding weights through stochastic gradient descent in a training scheme similar to tuning neural networks. The model can be formulated as

$$\widehat{y}_{ui} = f(q_u \cdot p_i + b_u + b_i) \tag{3}$$

$$f(x) = \frac{1}{1 + exp(-x)} \tag{4}$$

where q_u and p_i are user and item representations, b_u and b_i denote the bias term. $f(x)$ is a sigmoid function for binary prediction. We use Bayesian Personalised Ranking [14] pairwise loss, which is designed to maximize the prediction difference between a positive example and a randomly chosen negative example. It is useful when only positive interactions are present and the optimization of AUC is desired. We train the model with 30 epochs using the LightFM [15] implementation and generate two 10-dimensional latent vectors for user and item representations.

The aforementioned feature engineering results in a 160-dimensional $(2 \times (50 + 10 + 10 + 10))$ feature vector for each sample pair by concatenating all of above features.

3.2.3. Dot product

Even though the neural networks algorithm, such as multi-layer perceptron, can automatically learn an interaction function for computing relevant score, the experimental results shown in [11] demonstrate that a simple dot product (or element-wise product) operation is useful for making robust predictions. Therefore, we compute the dot product of each latent vector pair described in the previous section and generate 4-dimensional feature vector to express the relevance between user and item.

3.2.4. Statistic-based features

Given a specific categorical feature, the conditional probability of another one enriches information about users, items, and musics. To portray the user preference in an explicit way, we compute the conditional probability of a particular attribute given `user_id`. For example, we can derive which channel the user is prone to watch from $P(\texttt{channel}|\texttt{uid})$.

Likewise, for a continuous feature, we compute its conditional expectations given a specific `uid`. Moreover, the conditional standard deviation benefits the representability, too. For example, $E(\texttt{video\_duration}|\texttt{uid})$ and $\sigma(\texttt{video\_duration}|\texttt{uid})$ are adopted to describe the preference and habit of a user.

For this task, we use $P(\texttt{channel}|\texttt{uid})$, $P(\texttt{channel}|\texttt{item\_id})$, $P(\texttt{channel}|\texttt{music\_id})$, $E(\texttt{video\_duration}|\texttt{uid})$, $\sigma(\texttt{video\_duration}|\texttt{uid})$ to generate a 5-dimensional feature vector for each sample pair.

3.2.5. Title representation

Intuitively, the content of title has a noticeable influence on the click-through-rate. However, since only bag-of-words feature is provided, it's challenge to model the title's context, i.e., sequential relationship of words. In addition, the huge vocabulary size with 202,689 unique tokens make it inadvisable to represent the title using bag-of-words features. A low-memory alternative is to implement feature hashing based on the count-based feature, which finds the token string name to feature integer index mapping. In our solution, the number of features is set to be 2^4 and a 16-dimensional feature vector for each title representation is then created.

3.3. Deep Learning Based Framework

In this task, two sub-tasks are proposed: predicting the probability of `finish` and `like`. By exploring the distribution of the two prediction targets, we can find a positive value of `like` always followed by a positive value of `finish`; that is, modeling their dependency triggers the motivation of designing a multi-task deep learning framework.

The overview of the proposed neural network based model is illustrated in Figure 1. The sparse features, such as `uid` and `music_id`, first flow through an embedding layer for dense representation encoding. Then we leverage Compressed Interaction Network (CIN) [6] to model high-order feature interactions explicitly. Since CIN and multi-layer perceptron (MLP) take complementary advantages of each other, an intuitive way to enhance the model is to integrate these two structures. Moreover, for the multi-task learning scheme, two tasks share most of the hidden layers except for the final prediction layer.

As for the model inputs, we transform the sparse data into *multi-field* categorical form, which is widely used by deep neural networks based recommender model. An example is

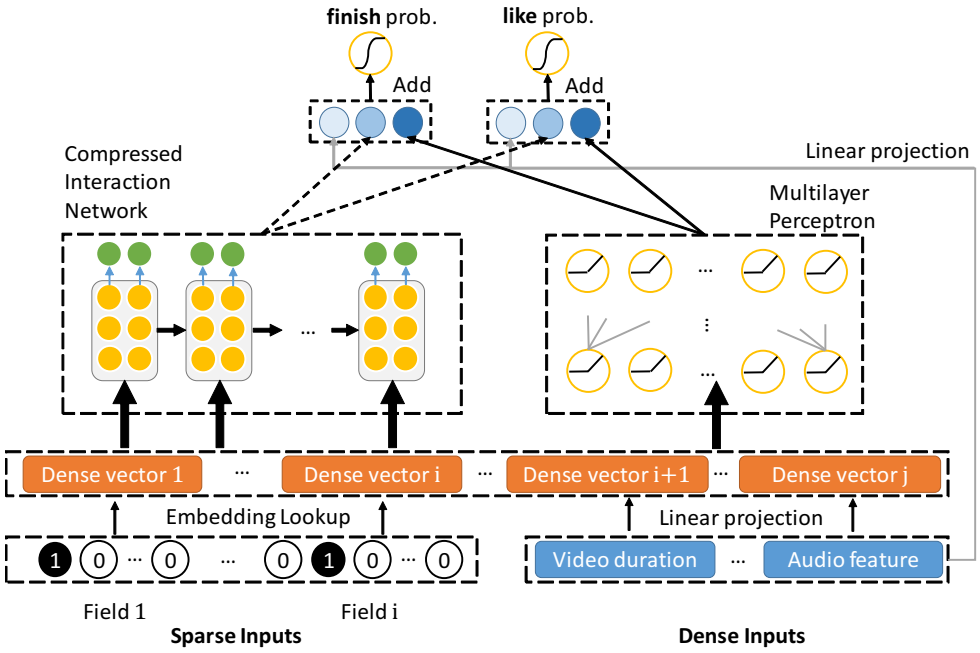

Fig. 1: The architecture of tailored neural networks with multi-task learning. Linear projection stands for $Linear(x) = x\mathbf{W}+b$.

shown below:

$$\underbrace{[0,1,0,...,0]}_{Field\ 1\text{: uid}} \quad \underbrace{[0,1,...,0]}_{Field\ 2\text{: user\_city}} \quad \underbrace{[1,0,...,0]}_{Field\ 3\text{: item\_id}} \quad \cdots \underbrace{[0,0,...,1]}_{Field\ i\text{: did}}$$

For dense inputs, such as `video_duration` and `audio_feature`, we first concatenate them and use fully connected layers to learn high-level feature representation.

After we convert all features into dense representations, both CIN and MLP are applied for learning. Moreover, a fully connected layer is used to learn dense input solo. Finally, three logits output from CIN, MLP, and dense inputs projection, are summed up together and the *sigmoid* function is responsible for the final output probability.

3.4. Gradient boosting decision tree

Although most winning models in data science competitions are ensembles of some advanced machine learning and deep learning algorithms, one particular model that is usually a part of such ensembles is the Gradient Boosting Decision Tree [16], which has quite a few effective implementations such as XGBoost and pGBRT. In this paper, we use LightGBM [17], a highly efficient gradient boosting decision tree implementation by Microsoft Research. The reason lies in that LightGBM accelerates the training process through Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB), and can achieve almost the same accuracy with far less time consuming, which is an important concern when dealing with tens of millions of data. The parameter settings would be discussed in the following sections.

4. EXPERIMENTS

4.1. Model training and parameters

Given the training set, we split 20% of the data for validation and parameter tuning. We update the neural networks based model by Adagrad [18] with a mini-batch size of 512 and train for number of epochs from one to three (depend on the number of features). Learning rate is set to 0.01. The $l2$ regularization is set to 0.00001. The setting for number of neurons per layer is 512 and 256 for both CIN and DNN layers; 128 for prediction layers. The activation functions in CIN and MLP are all ReLU [19].The dimension of field embedding is 8. The loss function is binary cross-entropy since the task is formulated as a binary classification problem.

In addition to NN-based model, two LightGBMs are trained separately for predicting the probability of `finish` and `like`, respectively. For LightGBM, we use a Scikit-learn API version of it and fine-tune the two most significant hyper-parameters, `n_estimators` and `num_leaves`, on validation set. The parameter `n_estimators` denotes the number of boosted trees to fit and `num_leaves` is the maximum tree leaves for base learners. Basically, we set `n_estimators` to 300 and 65 for the `finish` and `like` models, respectively, and `num_leaves` is set to 2048 and 4096. We leave the rest of parameters to default settings.

For the training procedure, even though the data is ordered chronologically, we find that shuffling training set before training could make the predictions more robust and the operation can be further treated as one kind of feature engi-

Table 2: Features descriptions

Code for features	Description	Dimension
a	categorical data	5,212,602
b	video_duration + create_time	2
c	conditional probability/expection features	5
d	concatenation of **user** and **item** latent feature from SVD	2×50
e	concatenation of **user** and **item** latent feature from LightFM	2×10
f	concatenation of **user** and **autho**r latent feature from SVD	2×10
g	concatenation of **user** and **music** latent feature from SVD	2×10
h	dot product feature	4
i	128-dimensional video features	128
j	128-dimensional audio features	128
k	title features	16

Table 3: Experimental results. (f) and (l) denote the predictions on finish and like, respectively.

Model	Feature	Private score	Public score (finish/like)
LightGBM(f) + NN(l)	a ,b ,c ,d ,e	0.7927	0.7918 (0.7331/0.9287)
LightGBM(f) + NN(l)	a ,b ,c ,d ,e ,i	0.7955	0.7945 (0.7364/0.9302)
4 LightGBMs(f) + 4 NNs(l)	a ,b ,c ,d ,e ,i	0.7955	0.7959 (0.7375/0.9321)
7 ensembles(f) + 5 ensembles(l)	a ,b ,c ,d ,e ,i	0.7987	0.7979 (0.7400/0.9328)
11 ensembles(f) + 7 ensembles(l)	a ,b ,c ,d ,e ,i ,g ,j	0.8008	0.7999 (0.7428/0.9332)
12 ensembles(f) + 8 ensembles(l)	a ,b ,c ,d ,e ,f ,g ,i ,j	0.8014	0.8005 (0.7436/0.9334)
17 ensembles(f) + 12 ensembles(l)	a ,b ,c ,d ,e ,f ,g ,i ,j ,k	0.8029	0.8020 (0.7455/0.9338)
24 ensembles(f) + 21 ensembles(l)	a ,b ,c ,d ,e ,f ,g, h, i, j, k	0.8038	0.8029 (0.7463/0.9348)

neering. With regard to the time consumption on training, it takes about 5 hours for NN models per epochs and 6 to 8 hours for LightGBM models. The machine we use is a GPU server with Xeon(R) E5-2667 v4, NVIDIA GeForce GTX 1080ti and 500 gigabytes of memory.

4.2. Evaluation results

In this subsection, we show our result on Leaderboard using different subset of features and various model settings in Table 3. The feature names are redefined in shorthand codes and illustrate in Table 2.

Based on our experiments on validation set, LightGBM provides better performance on predicting finish. On the contrary, the NN based model excels at predicting like. In Table 3, we demonstrate the evolution of our prediction results. The ensembles are a blend of LightGBMs and NNs with various training settings (e.g., more epochs and shuffling of data) by equally averaging the prediction probabilities. It is worth mentioning that more LightGBMs are used in ensembles(f) and more NNs are used in ensembles(l) following the same trend we've discovered. Moreover, we find that LightGBM and NN model are complementary since the prediction probabilities of them are quite uncorrelated. For example, the correlation coefficient of the predictions between a LightGBM and NN model using all features mentioned in Table 2 is 0.898 and 0.702 for finish and like, respectively. These are relatively low values since the AUC is approximate to 0.74 and 0.93 for the two sub-tasks, respectively. Besides, the reported feature importance by LightGBM indicates that the top significant features are **user** latent feature from LightFM, SVD, and video duration.

Our best submission achieves 0.8029 on the Public Leaderboard, and 0.8037 on the Private Leaderboard, which finished the 3^{rd} place in the competition.

5. CONCLUSION

In this paper, we detail our solution to the ICME2019 Short Video Understanding Challenge. We incorporate deep neural networks, which is equipped with Compressed Interaction Network, and Gradient Boosting Decision Tree for ensembles. The NN-based model is trained in a multi-task learning fashion, and we find that the two approaches surpass each other on the two sub-tasks. Feature engineerings, such as SVD and LightFM, are applied for performance boosting. With the aid of aforementioned techniques, we are able to achieve an AUC score of 0.8038 in the end and ranked the third in track2.

Due to memory constraints, we could not fit the Light-

GBM for track1 using the features we proposed. Note that it takes about 400 gigabytes of RAM to store the training matrix for track2. Therefore, in the future, a more efficient implementation to deal with the data structure and training procedure will be a vital issue.

Acknowledgments
This research was partially supported by Ministry of Science and Technology, Taiwan, under grant MOST-108-2634-F-002-008-.

6. REFERENCES

[1] Yehuda Koren, Robert Bell, and Chris Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 08, pp. 30–37, aug 2009.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv e-prints*, vol. abs/1409.0473, Sept. 2014.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[5] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He, "Deepfm: a factorization-machine based neural network for ctr prediction," *arXiv preprint arXiv:1703.04247*, 2017.

[6] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun, "xdeepfm: Combining explicit and implicit feature interactions for recommender systems," in *Proceedings of KDD '18*, New York, NY, USA, 2018, KDD '18, pp. 1754–1763, ACM.

[7] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang, "Deep & cross network for ad click predictions," in *Proceedings of the ADKDD'17*, New York, NY, USA, 2017, ADKDD'17, pp. 12:1–12:7, ACM.

[8] Steffen Rendle, "Factorization machines with libfm," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, pp. 57, 2012.

[9] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath, "The youtube video recommendation system," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, New York, NY, USA, 2010, RecSys '10, pp. 293–296, ACM.

[10] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami, "Mining association rules between sets of items in large databases," in *Acm sigmod record*. ACM, 1993, vol. 22, pp. 207–216.

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[12] Paul Covington, Jay Adams, and Emre Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM conference on recommender systems*. ACM, 2016, pp. 191–198.

[13] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, "Incremental singular value decomposition algorithms for highly scalable recommender systems," *Fifth International Conference on Computer and Information Science*, 01 2002.

[14] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 2009, pp. 452–461.

[15] Maciej Kula, "Metadata embeddings for user and item cold-start recommendations," in *Proceedings of ACM RecSys Workshop on New Trends on Content-Based Recommender Systems*. 2015, vol. 1448 of *CEUR Workshop Proceedings*, pp. 14–21, CEUR-WS.org.

[16] Jerome H Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[17] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, 2017, pp. 3146–3154.

[18] John Duchi, Elad Hazan, and Yoram Singer, "Adaptive subgradient methods for online learning and stochastic optimization," Tech. Rep. UCB/EECS-2010-24, EECS Department, University of California, Berkeley, Mar 2010.

[19] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.