An Auxiliary Task Boosted Multi-task Learning Method for Service Account Retrieval with Limited Human Annotation

Yuanzhou Yao [1,2], **Zhao Zhang** [1,2]*, **Kaijia Yang** [3], **Huasheng Liang** [3]
Qiang Yan [3] and **Yongjun Xu** [1,2]
[1] Institute of Computeing Technology, Chinese Academy of Sciences, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Tencent Wechat, Guangzhou, China
`{yaoyuanzhou21s, zhangzhao2021}@ict.ac.cn`
`{kogayang, watsonliang, rolanyan}@tencent.com`, `xyj@ict.ac.cn`

Abstract

Service accounts, including organizations' official accounts and mini-programs, provide various convenient services for users, and have become crucial components of a number of applications. Therefore, retrieving service accounts quickly and accurately is vital. However, this task suffers from the problem of limited human annotation, i.e., manually assessing account functionality and assigning ratings based on user experience is both labor-intensive and time-consuming. To this end, this paper proposes a novel approach, the Auxiliary task Boosted Multi-Task Learning method (AuxBoost-MTL). Specifically, the proposed method introduces multiple auxiliary tasks, which are able to utilized the log data from our application as supervision, and enhance the performance of the main task, service account retrieval. Furthermore, we introduce an Adaptive Hierarchical Fusion Module (AHF module) into our approach. This module is designed to adaptively perform hierarchical fusion of embeddings from auxiliary tasks into the main task, thereby enhancing the model efficacy. Experiments on two real-world industrial datasets demonstrate the effectiveness of our proposed approach.

1 Introduction

Service account retrieval services, exemplified by functionalities such as mini-program and official account searches, are fundamentally changing the way users interact with digital platforms (Hao et al., 2018). These accounts offer an array of services without the conventional need for downloading, proving critical in today's rapidly evolving digital age. However, despite their transformative potential and distinctive characteristics, service account retrieval services have not been thoroughly explored in the research field.

Contrary to traditional webpage retrieval that primarily involves query-document text matching (Rose and Levinson, 2004), service account retrieval provides functional services that cater to user queries, rather than merely facilitating simple text matching between queries and documents as seen in traditional webpage retrieval. This complexity presents significant challenges, particularly when annotating ranking datasets, a process that requires substantial time and effort due to the hands-on evaluation of each account's functionalities. The enormity of this task is underscored by our ongoing efforts, resulting in the annotation of approximately one hundred thousand entries in the service account search ranking dataset to date.

Addressing the urgent challenge of limited retrieval datasets, we propose an innovative Auxiliary task Boosted Multi-task Learning method (AuxBoost-MTL) that incorporates one main ranking task and four auxiliary tasks. Specifically, the proposed method is able to selectively fuse embeddings from the auxiliary tasks into the main task through an Adaptive Hierarchical Fusion Module (AHF module) to enhance training. Our application holds a vast amount of log data, where each data point (a query-account pair - numerous such pairs exist within a single search session) comprises a series of self-supervised labels readily available for training (Xie et al., 2021). In particular, we judiciously select four auxiliary tasks to correspondingly predict the following user feedback labels: click-through (Guo et al., 2017), retention (reuse within seven days), exit click (the last click within a search session), and dwell time (Kim et al., 2014). The first three tasks represent binary classification tasks, while the last task is a regression task.

AuxBoost-MTL offers two fundamental benefits for the main task of retrieval: 1) We take advantage of the rich log data from our application, which alleviate the issue of the dearth of annotated ranking datasets. Particularly, the availability of supervised signals from log data allow the shared layers of the multi-task framework to be trained, consequently

* Corresponding author: Zhao Zhang

enhancing the efficiency of the main ranking task (Ruder, 2017). 2) We introduce four auxiliary tasks to boost the main task. Specifically, with AHF module, the task-specific layers of the auxiliary tasks can be selectively incorporated into the main task's layer. This strategic integration refines the main task's embedding. AHF module achieves this by leveraging a knowledge graph for accounts and introducing the concept of "query entropy" to modulate the specificity and generality of a query.

We highlight the contributions as follows:

- To the best of our knowledge, our study pioneers research in service account retrieval systems, navigating its unique challenges and potentials.
- To address ranking data scarcity, we propose AuxBoost-MTL. AuxBoost-MTL employs extensive self-supervised log data and incorporates a bespoke module, AHF module, to enhance the main task-specific embedding.
- Experiments on two real-world datasets validate the efficacy of AuxBoost-MTL.

2 Related Work

2.1 Learning to Rank

Over the years, Learning-to-Rank (LTR) has become a focal area of research. The key objective is to train a scoring function that minimizes a ranking loss for the arrangement of documents (Liu et al., 2009). The field has evolved by improving ranking metrics through advancements in effective loss function design, transitioning from pointwise (Cao et al., 2007a) to pairwise (Jagerman et al., 2022) and then to listwise methods (Cao et al., 2007b). Model architectures have also seen significant development, from support vector machines (Joachims, 2002), gradient boosted decision trees (Burges, 2010; Wang et al., 2018), to the now prevalent neural networks (Burges et al., 2005, 2006; Li et al., 2019). They have demonstrated superior performance on conventional LTR datasets(Qin et al., 2021a,b). Neural ranking models, capable of handling large-scale datasets and diverse data types (Yates et al., 2021; Qin et al., 2021c), have been widely adopted in various industrial applications. Recently, Despite these advancements, ranking learning tasks under sparse data remain largely unexplored. To address this issue, we introduce a multi-task learning method, employing readily available user feedback as semi-supervised labels to enhance our ranking learning in sparse data scenarios.

2.2 Multi-task Learning

Significant strides have been made in the field of machine learning towards the application of multi-task learning (MTL) frameworks. Zhang et al. (Caruana, 1993) notably introduced an MTL framework that utilizes a shared bottom network to learn task-invariant representations, and task-specific networks to make predictions for individual tasks. This has inspired two main areas of development: enhancing task separation via constraints on task-specific parameters (Duong et al., 2015), and distinguishing shared from task-specific parameters(Ma et al., 2018b; Tang et al., 2020). These approaches have gained considerable attention in recommendation system (Pan et al., 2019; Lu et al., 2018) and webpage retrieval (Nishida et al., 2018; Sumbul and Demir, 2022). The notion of adaptive parameter sharing has gained traction, with methods like Task Adaptive Parameter Sharing (TAPS) offering a nuanced approach towards tuning a base model for new tasks by adaptively modifying a small, task-specific subset of layers, thus minimizing resource usage and competition among tasks (Wallingford et al., 2022). Similarly, understanding the relationships between tasks has been identified as crucial, as highlighted by studies focusing on the adaptive sharing of multi-level distributed representations (Wang et al., 2022). Furthermore, the advent of novel LSTM cells encapsulating both shared and task-specific parameters, as proposed in SC-LSTM (Lu et al., 2019), shows promise in improving the performance of a target task by judicious selection of auxiliary tasks, and hence, adds a new dimension to the ongoing discourse. Additionally, the reparameterization of convolutions for incremental multi-task learning has provided a pathway to manage task-specific parameters efficiently, especially when introducing new tasks to the MTL framework (Kanakis et al., 2020).

In our proposed model, we aim to encapsulate these advancements by integrating auxiliary task embeddings into the main task using our AHF module, thereby seeking to enhance both training and prediction performance. Through a meticulous amalgamation of shared and task-specific parameters, our model strives to strike a balance between task separation and parameter efficiency, while drawing inspiration from the aforementioned methodologies.

3 Problem Definition

We formalize our problem as a multi-task learning setting for service account retrieval. Given a query-account pair $x \in X$, we define $Y = \{y_r, y_c, y_u, y_e, y_t\}$ as the set of targets we aim to predict, where

- $y_r \in Y_r = \{0, 1, 2, 3, 4\}$ is the Relevance Level between the query and the account.
- $y_c \in Y_c$ is the Click-through: a binary indicator where 1 means the user clicked on the account, and 0 otherwise.
- $y_u \in Y_u$ is the Retention: a binary indicator where 1 means the user used the account again within seven days, and 0 otherwise.
- $y_e \in Y_e$ is the Exit Point: a binary indicator where 1 means the user exited the search session after clicking on the account, and 0 otherwise.
- $y_t \in Y_t$ is the Dwell Time: the time length that the user stayed on the account.

y_r is manually annotated, and the other four are obtained from log data. Predicting y_r is the main task, and predicting the other four labels/values are the auxiliary tasks. In particular, the log data is an array of query-account pairs' features and the associated user feedback. Formally, log data L can be represented as: $L = \{< q, a, F_q, F_a, F_{qa}, U > |q \in Q, a \in A\}$, where Q is the set of queries, A is the set of accounts, F_q, F_a and F_{qa} represent the features of queries, accounts and query-account pairs, respectively. The model's input is the combined feature set $F = F_q \cup F_a \cup F_{qa}$, while the output is the five labels/values $y = \{y_r, y_c, y_u, y_e, y_t\}$.

Objective: The goal is to minimize the following loss function:

$$L = \sum_{i \in \{r,c,u,e,t\}} \lambda_i \cdot L_i(y_i, f_i(F)) \qquad (1)$$

In this equation, $f_i(F)$ is the prediction function for task i, L_i is the corresponding loss for the task, and λ_i is the weight for the loss of task i. Our goal is to promote the performance of the main task.

4 Methodology

4.1 Overview

Our proposed AuxBoost-MTL model, shown in Figure 1, is purpose-built for service account retrieval and consists of three main components: the Shared Module, Task-specific Module, and the Adaptive Hierarchical Fusion Module. The Shared Module employs expert networks to analyze different aspects of input features and produces diverse output embeddings. The Task-specific Module then aggregates these embeddings based on their relevance to the main and auxiliary tasks and refines them using task-specific networks. Lastly, the Adaptive Hierarchical Fusion Module enhances the main task's performance by adaptively merging the auxiliary and main task-specific embeddings.

In the following sections, we delve deeper into each module's functionality and the training dynamics of the AuxBoost-MTL model.

4.2 Shared Module

The Shared Module is composed of a set of expert networks. Each expert network is designed to capture a distinct aspect of the input data. The input to the Shared Module consists of features from the query-account pair, denoted as $F = \{F_a, F_q, F_{qa}\}$. Let's define M as the total number of expert networks in the Shared Module. For each expert network m where $m \in \{1, 2, ..., M\}$, the network takes in the feature set F and outputs an embedding $\mathbf{e}_m$.

$$\mathbf{e}_m = Expert_m(F) \qquad (2)$$

Where $Expert_m(\cdot)$ denotes the m-th expert network in the Shared Module. The output of the Shared Module is the collection of these embeddings, denoted as $E = \{\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_M\}$. The embeddings in E capture different aspects of the input data, allowing the model to learn a comprehensive and diversified representation of the input.

4.3 Task-specific Module

In the Task-specific Module, each task is indexed by i where $i \in \{r, c, u, e, t\}$, maintains a set of learned weights, denoted as W_{im}. These weights determine the importance of each expert network's output to each task. The task-specific combined embedding $\mathbf{c}_i$ for each task is computed by a weighted combination of the expert embeddings, as follows:

$$\mathbf{c}_i = \sum_{m=1}^{M} W_{im} \cdot \mathbf{e}_m \qquad (3)$$

Where W_{im} signifies the weight of the m-th expert network for task i. Once the combined embeddings $\mathbf{c}_i$ for each task are derived, they are processed through respective task-specific tower networks.

Each tower network applies a series of transformation operations to the input, forming a task-specific embedding. Notably, the embeddings for the tasks of retention (u), exit point (e), and dwell time (t) are adjusted by incorporating the click-through task embedding, as these tasks are conditional on the click event. The task-specific embeddings $\mathbf{t}_i$ for all tasks are calculated as:

$$\mathbf{t}_i = \begin{cases} \mathbf{t}_c \cdot Tower_i(\mathbf{c}_i), & \text{if } i \in u, e, t, \\ Tower_i(\mathbf{c}_i), & \text{otherwise,} \end{cases} \tag{4}$$

where $Tower_i(\cdot)$ denotes the tower network associated with task i. The final set of task-specific embeddings $T = \{\mathbf{t}_r, \mathbf{t}_c, \mathbf{t}_u, \mathbf{t}_e, \mathbf{t}_t\}$ are directly used for prediction in the corresponding tasks and serve as the primary input to the Adaptive Hierarchical Fusion Module.

4.4 Adaptive Hierarchical Fusion Module

AHF module generates an auxiliary embedding for the main ranking task from two different perspectives: matching and quality. The matching embedding, indicated by the click-through rate task, gauges how well the account name matches the query regarding semantics. The intuition is that the user's click behavior comes from the matching degree between the query and the account name. The quality embedding, derived from a dynamic combination of three user feedback auxiliary task embeddings, represents the merit of an account in serving the query. This adaptive construction is implemented in two submodules: Quality Embedding Generation with Knowledge Graph, and Dynamic Fusion of Matching and Quality Embedding.

4.4.1 Quality Embedding Generation with Knowledge Graph

The first aspect of AHF module focuses on the generation of the quality embedding. It is worth noting that different accounts have varying weights from the three user feedback auxiliary tasks when it comes to creating the quality embedding. For example, for the one-off service like "report the loss of ID cards", the retention rate should have a relatively lower weight in the final quality embedding. Conversely, for game-type accounts, the retention rate and dwell time should be more significant in the final quality embedding as they reflect the account's quality. In order to adaptively adjust these weights depending on the query-account pair, we construct a knowledge graph for accounts and the corresponding organizations and companies, which contains about 10 million account entities. The knowledge graph is composed of three kinds of nodes, i.e., account, company/organization, and group/holding company, as well as three categories of edges. For example, the account "Meituan Grocery Shopping" belongs to the company "Beijing Baobao Love Food Catering Management Co., LTD.", and the company belongs to the group "Meituan-Dianping Group". This graph allows us to encode valuable facts about accounts, enhancing the retrieval task. To learn meaningful representations of this knowledge graph, We utilize the widely used knowledge graph embedding model DistMult (Yang et al., 2015) to learn the representations of entities and relations. DistMult uses a bilinear score function to compute the score of each knowledge triple (h, r, t), where h and t denote the head and tail entities, respectively, and r represents the relation between them. The score function is defined as

$$f_r(h, t) = \mathbf{h}^\top \mathbf{M_r} \mathbf{t}. \tag{5}$$

$\mathbf{h}$ and $\mathbf{t}$ denote the embeddings of h and t, respectively. $\mathbf{M_r}$ is a relation-specific diagonal matrix. The higher the score, the more likely the triple is true. The account embeddings obtained through this process serve as the gate embedding to guide the weight distribution of the three user feedback tasks in the final quality embedding.

Formally, we first obtain the embeddings $\mathbf{t}_u$, $\mathbf{t}_e$, and $\mathbf{t}_t$ from the retention, exit click, and dwell time tasks, respectively. We also compute the knowledge graph embedding $\mathbf{e}_g$ for each account using the DistMult model. The quality embedding $\mathbf{q}_v$ for an account is then calculated as follows:

$$\mathbf{q}_v = \sum_{i \in \{u,e,t\}} \left(\frac{\exp(\sigma(\mathbf{W}_g \cdot \mathbf{e}_g) \cdot \mathbf{t}_i)}{\sum_{j \in \{u,e,t\}} \exp(\sigma(\mathbf{W}_g \cdot \mathbf{e}_g) \cdot \mathbf{t}_j)} \right) \cdot \mathbf{t}_i \tag{6}$$

In this equation, $\mathbf{W}_g$ is a weight matrix, σ is the sigmoid function.

4.4.2 Dynamic Fusion of matching and Quality Embedding

In our task, queries are categorized into two types based on user intent, i.e., navigational queries and general requirement queries. Navigational queries are those where the user has a specific account in mind (for example, "Didi Taxi"). And general requirement queries are those where the user is expressing a need but does not specify a particular account (for example, "Hail a Taxi").

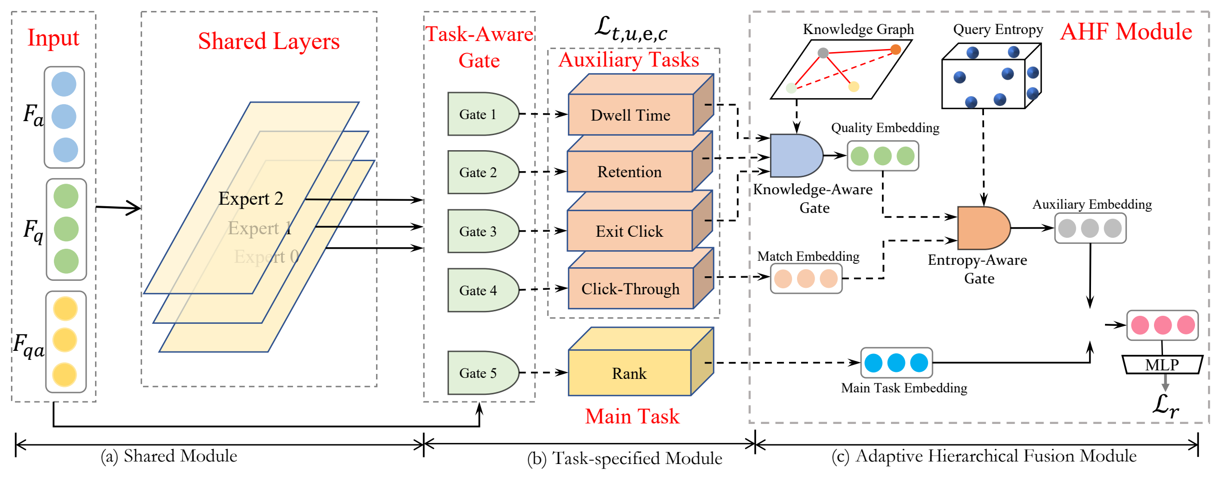

Figure 1: Overall architecture of AuxBoost-MTL

The key intuition behind this classification is that for navigational queries, the matching between the query and account is crucial, so the weight for the matching embedding should be high. For general requirement queries, the user is not committed to a specific account and is instead looking for the highest quality account that can satisfy his/her need. In this case, the quality embedding, which encapsulates the functionality and performance of the account, should be weighted more.

To dynamically adjust the weights of the matching and quality embeddings based on the query type, we introduce query entropy. The entropy of a query is a measure of the uncertainty or randomness in user clicks for each query. High entropy tends to signify a general requirement query, where user clicks are dispersed across multiple accounts. While low entropy indicates a navigational query, where most users click on a specific account. The query entropy is calculated as follows (Rényi, 1961):

$$H(q) = -\sum_{i=1}^{n} P(A_i^q|q) \log P(A_i^q|q) \tag{7}$$

where q denotes a query and A^q signifies the set of accounts that have been clicked under this query. The symbol n denotes the number of accounts in this set and $P(A_i^q|q)$ denotes the probability of clicking on account A_i^q given query q. The entropy $H(q)$ is a continuous feature and we discretize it into 100 bins. For each bin, we learn a corresponding gate embedding $\mathbf{e}_h$, which is used to calculate the weights for the matching and quality embeddings in the final auxiliary task embedding. The gate embedding is obtained via an embedding lookup: $\mathbf{e}_h = \text{Embedding}(H(q))$.

The final auxiliary embedding $\mathbf{a}_v$ is the softmax-weighted fusion of the matching embedding $\mathbf{m}_v$ and the quality embedding $\mathbf{q}_v$. This dynamic adjustment is mathematically expressed as:

$$\mathbf{a}_v = \sum_{k \in \{m,q\}} \frac{e^{\mathbf{e}_h \cdot \mathbf{k}_v}}{\sum_{k' \in \{\mathbf{m},\mathbf{q}\}} e^{\mathbf{e_h} \cdot \mathbf{k}'_\mathbf{v}}} \cdot \mathbf{k_v} \tag{8}$$

$\mathbf{e}_h$ represents the gate embedding, $\mathbf{k}_v$ stands for either the matching or the quality embedding.

4.4.3 Final Fusion and Loss Function

In the final phase, the auxiliary task embedding $\mathbf{a}_v$ is combined with the rank task-specific embedding $\mathbf{t}_r$ to generate a comprehensive embedding for each query-account pair:

$$\mathbf{z}_v = \text{MLP}(\mathbf{a}_v + \mathbf{t}_r) \tag{9}$$

Finally, we calculate the cross-entropy loss between the predicted labels $\hat{y} = \text{Softmax}(\mathbf{z}_v)$ and the true labels y, guiding the model to learn:

$$L_r = -\sum_{i=1}^{n} y_i \log(\hat{y}_i), \tag{10}$$

where n is the number of query-account pairs.

5 Experiments

5.1 Experimental Setups

5.1.1 Datasets

Since there is no existing dataset for account retrieval, we build two new datasets for official account retrieval and mini-program retrieval from

Table 1: Evaluation of various models on the mini-program and official-account datasets. "*" denotes the improvement is statistically significant compared with the best baseline at p-value < 0.05 over paired t-test.

	Model	Mini-Program				Official-Account			
		N@5	N@10	MRR	HR@1	N@5	N@10	MRR	HR@1
Single-Task	DeepFM	0.7388	0.8238	0.6121	0.6296	0.8422	0.8792	0.8824	0.8864
	DIFM	0.7502	0.9311	0.6177	0.6519	0.8558	0.8913	0.8975	0.9057
	AFN	0.7613	0.8402	0.6273	0.6808	0.8628	0.8964	0.8958	0.8999
	DCN	0.7637	0.8411	0.6333	0.6672	0.8512	0.8872	0.8947	0.9000
	FiBiNET	0.7761	0.8492	0.6342	0.6689	0.8628	0.8964	0.8957	0.9000
	OptFS	0.7834	0.8567	0.6469	0.7011	0.8534	0.8869	0.8899	0.8856
	AutoInt	0.7840	0.8554	0.6498	0.6995	0.8622	0.8965	0.8961	0.9024
	WDL	0.7872	0.8578	0.6534	0.7057	0.8633	0.8956	0.8934	0.9100
	NFM	0.7879	0.8577	0.6538	0.7057	0.8537	0.8866	0.8854	0.8799
	DCNMix	0.7827	0.8553	0.6472	0.7006	0.8438	0.8837	0.8927	0.8971
Multi-Task	Share-Bottom	0.7923	0.8612	0.6589	0.7045	0.8656	0.9016	0.8957	0.8997
	MMOE	0.7999	0.8624	0.6597	0.7187	0.8734	0.9037	0.9022	0.9089
	PLE	0.8022	0.8701	0.6689	0.7224	0.8712	0.8944	0.8956	0.9042
	MetaBalance	0.8054	0.8723	0.6842	0.7651	0.8788	0.9035	0.9012	0.9045
	AuxBoost-MTL	**0.8204***	**0.8811***	**0.7186***	**0.8092***	**0.8924***	**0.9207***	**0.9100***	**0.9254***

our application, namely Official-Account and Mini-Program, respectively. Precisely, we randomly select about one million search sessions from nearly one billion logs. Table 2 shows the detailed statistics of Official-Account and Mini-Program. In this table, $\#search$ indicates the number of search sessions, $\#annotated$ signifies the volume of manually annotated data, and $\#instance$ represents the total number of training data instances.

Table 2: Detailed statistics of two datasets.

Datasets	#search	#annotated	#instance
Official-Account	1,945,124	90,715	6,451,592
Mini-Program	875,451	63,534	3,708,455

5.1.2 Baseline and Metric

To compare the performance with state-of-the-art competitors, we select a total of 10 representative single-task models, including DeepFM (Guo et al., 2017), NFM (He and Chua, 2017), FiBiNET (Huang et al., 2019), DCN (Wang et al., 2017), WDL (Cheng et al., 2016), AFN (Cheng et al., 2020), DCNMix (Wang et al., 2021), AutoInt (Song et al., 2019), DIFM (Lu et al., 2021) and OptFS(Lyu et al., 2023). We also compare with four multi-task baselines, including Share-Bottom (Ruder, 2017), MMOE (Ma et al., 2018a), PLE (Tang et al., 2020) and MetaBalance (He et al., 2022). The models are evaluated using four well-established metrics:NDCG@5 (N@5), NDCG@10 (N@10) (Wang et al., 2013), Mean Reciprocal Rank (MRR), and Hit Ratio at Rank 1 (HR@1) (Handelman et al., 2018).

5.2 Overall Comparison

We evaluate various single-task and multi-task models on two datasets. The results, displayed in Table 1, yield the following insights:

- AuxBoost-MTL consistantly outperforms baselines, which indicates the effectiveness of the proposed method.
- Models exhibit superior performance on the Official-Account dataset compared to the Mini-Program dataset. This performance difference arises due to the larger volume of human-annotated data in the Official-Account dataset, underscoring the critical role such data plays in achieving optimal experimental outcomes.
- Additionally, across all evaluation metrics, multi-task models consistently outshine single-task ones. This pattern underlines the effectiveness of multi-task learning methods in leveraging and transferring information across tasks, leading to improved ranking performance.

5.3 Ablation Study

The results of our ablation study, as detailed in Table 4, validate the effectiveness of various components in our model. Without the incorporation of knowledge graph information when generating quality embeddings (w/o kg), the model shows a

Table 3: Case Study.

query	account name	weight assignment in gate mechanism
DiDi Taxi	DiDi Taxi	**Quality Embedding:** Exit Click (0.3055), Retention (0.6606), Dwell Time (0.0338) **Auxiliary Embedding:** matching Embedding (0.6587), Quality Embedding (0.3413)
Hail a Taxi	DiDi Taxi	**Quality Embedding:** Exit Click (0.3047), Retention (0.6570), Dwell Time (0.0383) **Auxiliary Embedding:** matching Embedding (0.2886), Quality Embedding (0.7114)
Two-player Game	Overcooked	**Quality Embedding:** Exit Click (0.2367), Retention (0.2459), Dwell Time (0.5174) **Auxiliary Embedding:** matching Embedding (0.3184), Quality Embedding (0.6816)

Table 4: Ablation tests on Mini-Program.

	N@5	N@10	MRR	HR@1
Our	**0.8204**	**0.8811**	**0.7186**	**0.8092**
w/o kg	0.8186	0.8796	0.7129	0.8092
w/o entropy	0.8172	0.8789	0.7137	0.7996
w/o hierarchical	0.8155	0.8742	0.7088	0.7732
w/o auxiliary	0.8136	0.8739	0.7052	0.7693

minor decline in all performance metrics, highlighting the knowledge graph's contribution to the model's efficacy. Similarly, when we disregard the role of query entropy in shaping the auxiliary embedding (w/o entropy), the slight drop in scores further underscores the importance of entropy in refining the model's performance. Most notably, the complete removal of auxiliary embedding (w/o auxiliary) results in a much more substantial decrease in performance, emphatically demonstrating the vital role of auxiliary tasks in our method. The ablation study thus proves that each component of our model plays a critical role in enhancing the overall effectiveness of service account retrieval.

5.4 Case Study

In our case studies (Table 3), we scrutinize various query-account pairs to highlight the adaptability and efficiency of our gate mechanism in weight assignments. For a functional account like "DiDi Taxi", user retention consistently receives the highest weight (66.06% and 65.70%) in the Quality Embedding, indicating its primary assessment method. But for "Overcooked", a popular multiplayer game, Dwell Time becomes the prevailing factor (51.74%), reflecting the importance of user engagement duration in measuring the quality of game accounts. These examples show how our model, armed with knowledge graph information, adapts weight distribution in Quality Embedding. Further examination of the Auxiliary Embedding weights reveals differences in user's intents. For a navigational query like "DiDi Taxi", matching Embedding receives more weight (65.87%), signaling a clear user preference for the account. Conversely, for a general requirement query like "Hail a Taxi", Quality Embedding outweighs (71.14%), indicating the model's preference for high-quality taxi services over a specific account.

6 Conclusion

This paper introduces a novel method, AuxBoost-MTL, for service account retrieval. AuxBoost-MTL addresses the challenge of sparse annotated data by utilizing user feedback in abundant log data, and trains auxiliary tasks through self-supervision. Specifically, the proposed method consists of three modules. In particular, the first two modules learn task-specific embeddings, and the third module, the AHF module, is able to adaptively fuses embeddings from auxiliary tasks into the main ranking task. Experimental results on real-world datasets demonstrate the effectiveness of AuxBoost-MTL, making it a promising solution for service account retrieval systems.

Acknowledgements

The research work is supported by the National Natural Science Foundation of China under Grant No. 62206266, and the 2022 Tencent Wechat Rhino-Bird Focused Research Program.

References

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96.

Christopher Burges, Robert Ragno, and Quoc Le. 2006. Learning to rank with nonsmooth cost functions. *Advances in neural information processing systems*, 19.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007a. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007b. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136.

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *International Conference on Machine Learning*.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10.

Weiyu Cheng, Yanyan Shen, and Linpeng Huang. 2020. Adaptive factorization network: Learning adaptive-order feature interactions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3609–3616.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (volume 2: short papers)*, pages 845–850.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. Deepfm: A factorization-machine based neural network for ctr prediction. *international joint conference on artificial intelligence*.

Guy S Handelman, Hong Kuan Kok, Ronil V Chandra, Amir H Razavi, Shiwei Huang, Mark Brooks, Michael J Lee, and Hamed Asadi. 2018. Peering into the black box of artificial intelligence: Evaluation metrics of machine learning methods. *AJR. American journal of roentgenology*, 212(1):38–43.

Lei Hao, Fucheng Wan, Ning Ma, and Yicheng Wang. 2018. Analysis of the development of wechat mini program. In *Journal of Physics: Conference Series*, volume 1087, page 062040. IOP Publishing.

Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 355–364.

Yun He, Xue Feng, Cheng Cheng, Geng Ji, Yunsong Guo, and James Caverlee. 2022. Metabalance: improving multi-task recommendations via adapting gradient magnitudes of auxiliary tasks. In *Proceedings of the ACM Web Conference 2022*, pages 2205–2215.

Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. Fibinet: combining feature importance and bilinear feature interaction for click-through rate prediction. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 169–177.

Rolf Jagerman, Zhen Qin, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2022. On optimizing top-k metrics for neural ranking models. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2303–2307.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142.

Menelaos Kanakis, David Bruggemann, Suman Saha, Stamatios Georgoulis, Anton Obukhov, and Luc Van Gool. 2020. Reparameterizing convolutions for incremental multi-task learning without task interference. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pages 689–707. Springer.

Youngho Kim, Ahmed Hassan, Ryen W White, and Imed Zitouni. 2014. Modeling dwell time to predict click-level satisfaction. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 193–202.

Pan Li, Zhen Qin, Xuanhui Wang, and Donald Metzler. 2019. Combining decision trees and neural networks for learning-to-rank in personal search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2032–2040.

Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331.

Peng Lu, Ting Bai, and Philippe Langlais. 2019. SC-LSTM: Learning task-specific representations in multi-task learning for sequence labeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2396–2406, Minneapolis, Minnesota. Association for Computational Linguistics.

Wantong Lu, Yantao Yu, Yongzhe Chang, Zhen Wang, Chenhui Li, and Bo Yuan. 2021. A dual input-aware factorization machine for ctr prediction. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3139–3145.

Yichao Lu, Ruihai Dong, and Barry Smyth. 2018. Why i like it. *Proceedings of the 12th ACM Conference on Recommender Systems*.

Fuyuan Lyu, Xing Tang, Dugang Liu, Liang Chen, Xiuqiang He, and Xue Liu. 2023. Optimizing feature set for click-through rate prediction. *arXiv preprint arXiv:2301.10909*.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018a. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1930–1939.

Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018b. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1137–1140.

Kyosuke Nishida, Itsumi Saito, Atsushi Otsuka, Hisako Asano, and Junji Tomita. 2018. Retrieve-and-read: Multi-task learning of information retrieval and reading comprehension. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 647–656, New York, NY, USA. Association for Computing Machinery.

Junwei Pan, Yizhi Mao, Alfonso Lobos Ruiz, Yu Sun, and Aaron Flores. 2019. Predicting different types of conversions with multi-task learning in online advertising. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Zhen Qin, Le Yan, Yi Tay, Honglei Zhuang, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2021a. Born again neural rankers.

Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Mike Bendersky, and Marc Najork. 2021b. Are neural rankers still outperformed by gradient boosted decision trees?

Zhen Qin, Honglei Zhuang, Rolf Jagerman, Xinyu Qian, Po Hu, Dan Chary Chen, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2021c. Bootstrapping recommendations at chrome web store. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3483–3491.

Alfréd Rényi. 1961. On measures of entropy and information. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, volume 4, pages 547–562. University of California Press.

Daniel E Rose and Danny Levinson. 2004. Understanding user goals in web search. In *Proceedings of the 13th international conference on World Wide Web*, pages 13–19.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1161–1170.

Gencer Sumbul and Begüm Demir. 2022. Plasticity-stability preserving multi-task learning for remote sensing image retrieval. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–16.

Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 269–278.

Matthew Wallingford, Hao Li, Alessandro Achille, Avinash Ravichandran, Charless Fowlkes, Rahul Bhotika, and Stefano Soatto. 2022. Task adaptive parameter sharing for multi-task learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7561–7570.

Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, pages 1–7.

Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the Web Conference 2021*, pages 1785–1797.

Tianxin Wang, Fuzhen Zhuang, Ying Sun, Xiangliang Zhang, Leyu Lin, Feng Xia, Lei He, and Qing He. 2022. Adaptively sharing multi-levels of distributed representations in multi-task learning. *Information Sciences*, 591:226–234.

Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2018. The lambdaloss framework for ranking metric optimization. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 1313–1322.

Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. 2013. A theoretical analysis of ndcg type ranking measures. In *Conference on learning theory*, pages 25–54. PMLR.

Ruobing Xie, Cheng Ling, Yalong Wang, Rui Wang, Feng Xia, and Leyu Lin. 2021. Deep feedback network for recommendation. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 2519–2525.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*.

Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. 2021. Pretrained transformers for text ranking: Bert and beyond. In *Proceedings of the 14th ACM International Conference on web search and data mining*, pages 1154–1156.