# Research: AMinerGNN: Heterogeneous Graph Neural Network for paper click-through rate prediction with fusion query

Anonymous Author(s)

## ABSTRACT

Paper recommendation with user-generated keyword is to suggest papers that simultaneously meet user's interests and are relevant to the input keyword. This is a recommendation task with two queries, a.k.a. user ID and keyword. However, existing methods focus on recommendation according to one query, a.k.a. user ID, and are not applicable to solving this problem. In this paper, we propose a novel click-through rate (CTR) prediction model with heterogeneous graph neural network, called AMinerGNN, to recommend papers with two queries. Specifically, AMinerGNN constructs a heterogeneous graph to project user, paper, and keyword into the same embedding space by graph representation learning. To process two queries, a novel query attentive fusion layer is designed to recognize their importances dynamically and then fuse them as one query to build a unified and end-to-end recommender system. Experimental results on our proposed dataset and online A/B tests prove the superiority of AMinerGNN.

## CCS CONCEPTS

• **Information systems** → **Recommender systems**.

## KEYWORDS

Click-Through Rate Prediction, Graph Neural Network, Recommender System

**ACM Reference Format:**

Anonymous Author(s). 2022. Research: AMinerGNN: Heterogeneous Graph Neural Network for paper click-through rate prediction with fusion query. In *Proceedings of 31st ACM International Conference on Information and Knowledge Management (CIKM 2022).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/1122445.1122456

## 1 INTRODUCTION

AMiner[1] is an academic information retrieval website, which can provide paper recommendation to researchers. Depending on whether the user inputs a keyword or not, there are two recommender systems (RSs) for online services, called General-RS and Keyword-RS (as shown in Figure 1). When simultaneously given the user ID and a user-generated keyword (usually a research direction, such as *graph neural network* or *pretraining*), Keyword-RS suggests papers that not only are relevant to this keyword but also satisfy user potential interests.

According to the relation between the user-generated keyword and the research direction of the user based on his interacted papers, there are three segmentation scenarios in Keyword-RS: (1) **both of them refer to the same research field (S1)**; (2) **there exists the potential of interdisciplinary research between them (S2)**; (3) **they are independent and have no relevance (S3)**. We collect some user feedbacks and have an important finding: **the purpose that researchers use AMiner and the importance of these two queries vary according to scenarios**, which are as follows: (1) In **S1**, users aim to make a deeper literature review about this research field, like tracking the advanced methods or seeking the highly cited papers. At this time, Keyword-RS should utilize his interests to recommend and user ID is more important than keyword since user ID contains the personal information. (2) In **S2**, users intend to know how this approach (user-generated keyword) can be integrated into his past research, like how contrastive learning can be used for recommendation. Two queries are both useful in this situation. (3) In **S3**, users prefer to improve the general understanding of this new field (we call it a new field since this keyword never appears in his interacted papers), like Bert [4] in the field of Natural Language Processing. Therefore, it is reasonable to recommend papers without personal preferences and show some typical or hot papers clicked by most researchers, which means the keyword is more important than user ID. To satisfy the above recommendation requirements, there are two problems: (1) **how to model the relation between the user-generated keyword and his past research direction**; (2) **how to build a unified and end-to-end recommender system applied in all scenarios**.

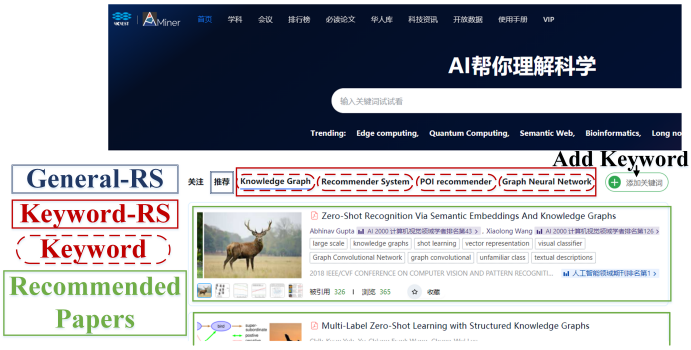


**Figure 1: Keyword-RS in AMiner. Best viewed in color.**

The above business is a typical click-through rate (CTR) prediction task but with a novel input, a.k.a. user ID and keyword. Early works predict CTR via automatic feature interactions like DeepFM [9], PNN [21] and InterHAt [17]. Recent researchers aims to model user behaviors (e.g. DIN [28], UBR [20] and SIM [19]) and time-varying inerests (e.g. DIEN [27] and SURGE [3]). However, all of them are given just one query, a.k.a. user ID, and are less applicable. Another close solution is the methods in product search by regarding the input keyword as the search query. They focus on infering personalized search inclination (e.g. HEM [1] and GEPS[26] ) or analysing users' long and short-term preferences (e.g. ALSTP [11] and MGDSPR [16]). However, they neglect that the importances of personalization and search-aware intention are dynamically changing with scenarios.

To address the foregoing problems, we propose an end-to-end heterogeneous graph neural network for paper click-through rate (CTR) prediction, called AMinerGNN. It consists of two components: (1) **Graph embedding layer**. To model the semantic correlation in Keyword-RS, we construct a heterogeneous graph, which consists of user, paper and keyword, and project the above three types of entities into the same embedding space via graph based representation learning, which is the basic to further learn the relation between user-generated keyword and his past research direction. (2) **Query attentive fusion layer**. We employ an attention unit to automatically mine user scenario-specific purpose and dynamically adjust the importances of two queries, which are further combined as one query to build a unified recommender system.

## 2 RELATED WORKS

We briefly summarize two related subareas of information retrieve systems, which are click-through rate prediction and product search, as follows.

Click-through rate (CTR) estimation plays as a core function module in various personalized online services. Feature interaction [10, 12, 21, 22] is an early method to build a predictive model. FM [22] assigns a k-dimensional learnable embedding vector to each feature and explores their 2-order interactions. DeepFM [10] incorprates the deep network to learn high-order feature interactions effectively. In recent years, modeling user behaviors [3, 19, 20, 27, 28] is becoming an essential topic since they

[1] www.aminer.com

*CIKM 2022, October 17-22, 2022, Atlanta, Georgia, USA*
2018. ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

contain crucial patterns of user interests. DIN [28] introduces an attention mechanism to attribute different historical behaviors with different scores according to the relatedness with the target item. Based on DIN, DIEN [27] utilizes GRU structure to capture the temporal interests. UBR [20] proposes a new framework that uses search engine techniques to retrieve the most relevant behaviors, which not only solves the time complexity problem but also alleviates the noisy signals in long consecutive sequences. SURGE [3] performs cluster-aware and query-aware graph convolutional propagation to extract users' current activated core interests from long but noisy behavior sequences. However, their inputs are userID and target item, which can not directly resolve the CTR problem with fusion query.

Product search is an important way for people to browse and purchase items on E-commerce platforms. A basic category of methods [5–7, 13] is to associate the free-form user queries with structured data stored in relational databases. MAM [6] proposes a probabilistic retrieval model to mine useful knowledge from product search log data. [13] discusses issues and strategies when introducing learning-to-rank (LETOR) methods to the product search. Recent works aim to study the user's personal and temporal preferences [1, 2, 11, 16, 26]. HEM [1] attentively combines userID and query to predict personal behavior by mapping the user into the same latent space with the query and product. ALSTP [11] integrates the long and short-term user interests with the current query for the personalized product search. MGDSPR [16] simultaneously models query semantics and historical behaviors, aiming at retrieving more products with good relevance. However, all of them are under-explored in the task where the search query and personal preferences have non-uniform relations.

# 3 PRELIMINARIES

In this section, we first formulate the problem and then give a detailed description of the construction of the heterogeneous graph used in Keyword-RS.

## 3.1 Problem Setup

**Paper CTR with two queries**. Given a set $< \mathcal{U}, \mathcal{K}, \mathcal{P} >$, where $\mathcal{U} = \{u_1, \cdots, u_M\}$ denotes $M$ users, $\mathcal{K} = \{k_1, \cdots, k_T\}$ denotes $T$ keywords and $\mathcal{P} = \{p_1, \cdots, p_N\}$ denotes $N$ papers. In this problem, user and keyword are both regarded as queries, leading to three kinds of interaction datas, which are user-paper, keyword-paper and user&keyword-paper interactions. The above three interactions are represented as $\mathcal{Y} = \{y_{u-p}, y_{k-p}, y_{uk-p} | u \in \mathcal{U}, k \in \mathcal{K}, p \in \mathcal{P}\}$, respectively. $y_{uk-p}$ indicates whether $u$ clicks $p$ when $u$ inputs keyword $k$. $y_{u-p}$ means whether $u$ clicks $p$ in Keyword-RS or searches $p$ in AMiner. $y_{k-p}$ represents whether $p$ is clicked when the input keyword is $k$. Note that $y_{u-p}$ leverages both search and recommendation log while the other two just use recommendation log. We also utilize two widely used information [15, 18, 25] to improve CTR performance: (1) query behaviors including two aspects: user behaviors $H_u = \{p | y_{u-p} = 1\}$ and keyword behaviors $H_k = \{p | y_{k-p} = 1\}$, (2) item features $F^p = [F^p_1, \cdots, F^p_q, \cdots]$, where $F^p_q$ denotes $q$-th feature of paper $p$ like citation number or publication year. Note that $H_u$ reckons without keywords and represents papers that $u$ clicked, while $H_k$ denotes papers all users clicked when the input keyword is $k$. When user $u$ inputs a keyword $k$, the Keyword-RS is to required to prediect whether $u$ will click $p$ as $\hat{y}_{uk-p} = f(u, k, p, H_u, H_k, F_p)$.

## 3.2 Graph Construction

We construct a heterogenous graph $\mathcal{G}$ to hold the semantic relatedness between different entities in Keyword-RS. Specifically, there are three types of nodes: user, paper, and keyword. We split the title of the paper into individual words[2] and filter out stopwords by word tokenization tool like nltk[3], and then gather these words as the keyword set $\mathcal{K}$. Then three edge types are built from raw interactions between nodes as follows:

- **user-paper**. When user $u$ clicks or searches paper $p$, an edge $(u, p)$ is added.
- **user-keyword**: When user $u$ adds keyword $k$ as shown in figure 1, an edge $(u, k)$ is added.
- **paper-keyword**: When the title of paper $p$ contains the keyword $k$, an edge $(p, k)$ is added.

In summary, $\mathcal{G}$ is defined as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, O_\mathcal{V}, O_\mathcal{E}\}$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$. And $O_\mathcal{V} = \{\mathcal{U}, \mathcal{P}, \mathcal{K}\}$ and $O_\mathcal{E} = \{u-p, u-k, p-k\}$ denote the set of node types and that of edge types. Note that all edges indicate a symmetric relation, which means $\mathcal{G}$ is an undirected graph. Figure 2(a) shows an example of the above three situations.

# 4 METHODOLOGY

In this section, we start from the motivations and intuitions, and then describe the technical details of the proposed AMinerGNN, which follows a hierarchical structure: GNN embedding layer → query attentive fusion layer. Figure 2 presents the framework of AMinerGNN.

## 4.1 Motivation

To tackle two problems mentioned in Section 1, a three-step solution is designed as follows: (1) representing user, paper, and keyword in the same embedding space → (2) enhancing representations via raw interactions between entities → (3) feeding user and keyword representations into a unified CTR model. The first two steps and last step are implemented in Graph embedding Layer (cf. Section 4.2) and query attentive fusion layer (cf. Section 4.3), respectively.

Note that GNN is of great significance. Specifically, there are three advantages on different entity sides: (1) **On the paper side**. Paper representations, which keywords are integrated into, can contain the information of the research direction of the paper. (2) **On the user side**. The interacted papers of a user represent his past research direction and preferences in viewing papers. What's more, the keywords user adds directly reflect his concerned research field. Therefore, it is pivotal to enhance user representations with the above two entities. (3) **On the keyword side**. If two keywords co-occur in one paper, it shows the potential of interdisciplinary research between them. Thus, mutually enhanced representations of a pair of keywords can bring them closer in latent semantic space. The closer the distance between two keyword embeddings, the greater the potential of interdisciplinary research.

## 4.2 Graph embedding Layer

To achieve the first two steps, GNN is an intuitive tool to enhance node representation using multi-hop neighbors' information. In each layer, we employ a widely used two-step scheme[8, 23, 24] to aggregate neighbors: (1) same relation-aware neighbors aggregation via average pooling; (2) relations combination via sum pooling. And we leave the further work of other pooling methods like attention or metapath based aggregation as the future work. More formally, in the $l$-th layer, node representation is updated as

$$\mathbf{e}_i^{(l)} = \sum_{t \in O_\mathcal{E}} \frac{1}{\mid \mathcal{N}_i^t \mid} \sum_{j \in \mathcal{N}_i^t} \mathbf{e}_j^{(l-1)} \tag{1}$$

where $\mathbf{e}_i^{(l)}$ denotes the embedding of node $i$ in the $l$-th layer and $\mathbf{e}_i^{(0)}$ is randomly initialized. Here $i$ can be one of the user, paper or keyword node. $\mathcal{N}_i^t$ denotes the $t$-type neighbors for node $i$ and $t \in O_\mathcal{E}$.

After performing $L$ layers, we obtain multiple node representations and then adopt sum pooling to integrate multi-hop information as

$$\mathbf{e}_i^* = \sum_{l=0}^{L} \mathbf{e}_i^{(l)} \tag{2}$$

where superscript $*$ indicates that the representation is enhanced by GNN.

## 4.3 Query Attentive Fusion Layer

With all representations in the same embedding space, we calculate the correlation between a user and a keyword, denoted as $\gamma_{uk}$, using distance

[2]Here we use words in the title rather than keywords, the reason is that there are obstacles in obtaining the keyword information, which is absent in a part of papers.
[3]https://github.com/nltk/nltk

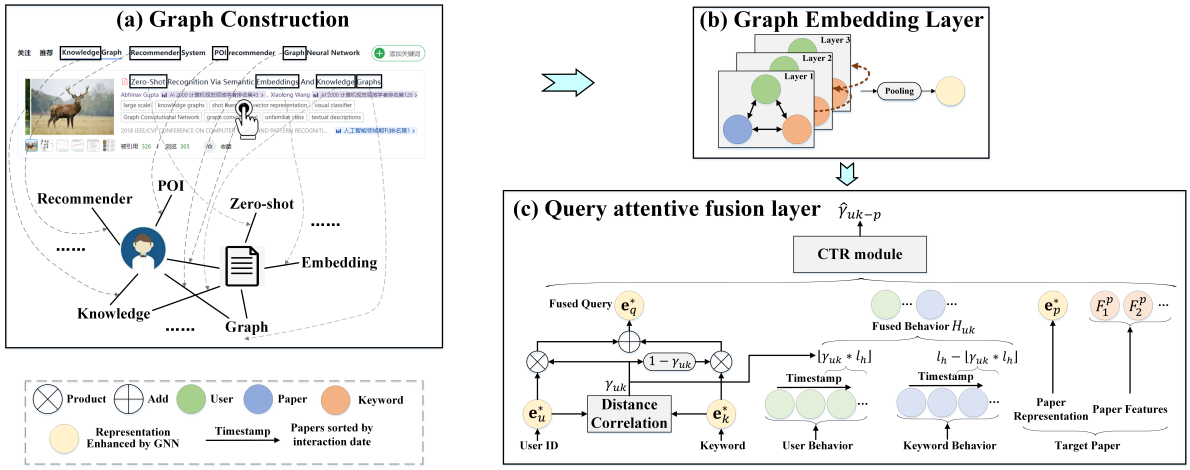


**Figure 2: The framework of AMinerGNN. Best viewed in color.**

correlation as

$$\gamma_{uk} = d\,\mathrm{Cor}\left(\mathbf{e}_u^*, \mathbf{e}_k^*\right) = \frac{d\,\mathrm{Cov}\left(\mathbf{e}_u^*, \mathbf{e}_k^*\right)}{\sqrt{d\,\mathrm{Cov}\left(\mathbf{e}_u^*, \mathbf{e}_u^*\right) \cdot d\,\mathrm{Cov}\left(\mathbf{e}_k^*, \mathbf{e}_k^*\right)}} \tag{3}$$

where $d\,\mathrm{Cov}(\cdot)$ is the distance covariance of two representations.

As aforementioned in Section 1, the relation between two queries is non-uniform for all scenarios. More specifically, **the importance of user ID is in proportion to the correlation between two queries**. Therefore, an intuitive idea is regrading correlation $\gamma_{uk}$ as the attention weight to form a unified and fused query as

$$\mathbf{e}_q^* = \gamma_{uk}\mathbf{e}_u^* + (1-\gamma_{uk})\mathbf{e}_k^* \tag{4}$$

Note that $\mathbf{e}_q^*$ can represent user purposes in different scenarios. For example, consider the limit case, when $\gamma_{uk} = 0$ which means the user-generated keyword is absolutely unrelated to his past research direction, $\mathbf{e}_q^* = \mathbf{e}_k^*$ is reasonable because there is no need to use personal interest for recommendation at this situation. It has similar rationality in the other two scenarios.

After two queries are fused, another problem is how to process query behaviors $H_u$ and $H_k$. Similar to the fusion of $\mathbf{e}_u^*$ and $\mathbf{e}_k^*$, a heuristic idea is to combine a partial sequence of each behavior as one new behavior, denoted as $H_{uk}$. The length of each partial behavior is controlled by $\gamma_{uk}$. For example, when $\gamma_{uk}$ is large, we hope to reserve more user behaviors and less keyword behaviors, and vice versa. For faster and unified tensor calculation, we set the length of $H_{uk}$ as a fixed number $l_h$. Then $\lfloor \gamma_{uk} * l_h \rfloor$ recent papers in $H_u$ and $l_h - \lfloor \gamma_{uk} * l_h \rfloor$ recent papers in $H_k$ are selected to form $H_{uk}$. The notation $\lfloor x \rfloor$ refers to the greatest integer that is less than or equal to $x$. Formally, $H_{uk}$ is generated as

$$H_{uk} = \{p_1, p_2 | p_1 \in H_u, p_2 \in H_k, |p_1| = \lfloor \gamma_{uk} * l_h \rfloor = l_h - |p_2|\} \tag{5}$$

Finally, we fed fused query $\mathbf{e}_q^*$ and bahavior $H_{uk}$, and other auxiliary information $F^p$ into a CTR module to predict the probability of paper $p$ being clicked as

$$\hat{y}_{uk-p} = f_{\mathrm{CTR}}(\mathbf{e}_q^*, \mathbf{e}_p^*, H_{uk}, F_p) \tag{6}$$

where $f_{\mathrm{CTR}}$ can be any CTR model, which is set as DIEN [27] by default.

Note that AMinerGNN can be generalized to CTR prediction with multi queries. For example, when the user inputs two keywords, the idea of query fusion is still productive to combine three queries (one user ID and two keywords) as one attentively, and three attention weights are distributed according to their importances, which we leave as future work.

**Table 1: Statistics of the AMiner_KRS dataset. 'u', 'k' and 'p' denote user, keyword and paper, respectively.**

| | | | | |
|---|---|---|---|---|
| Graph $\mathcal{G}$ | node | # users | # keywords | # papers |
| | | 67,806 | 398,001 | 200,837 |
| | edge | # u-p | # u-k | # p-k |
| | | 568,424 | 240,116 | 6,899,284 |
| Interactions $\mathcal{Y}$ | | # $y_{u-p}$ | # $y_{k-p}$ | # $y_{uk-p}$ |
| | | 568,424 | 5,030 | 2,781 |

## 4.4 Model Optimization

We use Logloss, which is the most used objective function in CTR prediction, to capture divergence between two probability distributions as

$$\mathcal{L} = -\frac{1}{N}\sum_{(u,k,p)\in S}\left(y_{uk-p}\log\hat{y}_{uk-p} + (1-y_{uk-p})\log(1-\hat{y}_{uk-p})\right) \tag{7}$$

where $S$ is the training set of size $N$.

# 5 EXPIREMENTS

## 5.1 Experimental Settings

*5.1.1 Dataset Description.* We propose a novel dataset, called AMiner_KRS dataset. We collect the interaction data in the past three months, from Feb. 1, 2022, to May 1, 2022. To ensure data quality, we filter the unregistered users and low-quality papers[4]. Finally, AMiner_KRS consists of 67806 users, 398001 papers, and 200837 keywords.To avoid data leakage, we use the former two months' data to construct the graph and generate query behaviors, while the user&keyword-paper interaction data in the third month (Apr. 2022) is randomly split into training (80%) and testing (20%) sets. Table 1 lists the statistics of the AMiner_KRS dataset.

*5.1.2 Baselines.* To demonstrate the effectiveness, we compare AminerGNN with two kinds of methods: (1) **CTR** models including DeepFM [10], DIN [28], DIEN [27], UBR [20]), (2) **Product search** methods including HEM [1], ALSTP [11], MGDSPR [16].

*5.1.3 Experimental Settings.* The core contribution of AMinerGNN is generalizing traditional CTR models to the task with fusion query using two key designs: (1) GNN embedding layer and (2) query fusion layer. Therefore, our experiments aim to prove the effectiveness of these two parts. Specifically, for base model, two queries are added to one embedding and two behaviors are fused into an union set, which are directly fed into CTR model, such as $f_{\mathrm{DIN}}(\mathbf{e}_u + \mathbf{e}_k, \mathbf{e}_p, H_u \cup H_k, F_p)$. Then we compare each model with the following three variants: (1) using graph embedding

[4] AMiner has collected about billions of papers stored in MongoDB database. However, most of them are viewed by researchers on our website less than 5 times. Therefore, we just retain the high-quality papers by setting the threshold of citation number and view number.

layer, (2) using query attentive fusion layer, and (3) both adopting these two layers. We use subscript "base", "g", "f" and "g&f" to denote the base model and corresponding three kinds of variants. Note that AMinerGNN is identical to $\text{DIEN}_{\text{g\&f}}$.

The schema of input in product search methods usually has three sequences, which are users, corresponding queries, and corresponding products. We use $y_{uk-p}$ to generate the above three sets. For ALSTP, we search the number of previous queries during short-term preference modeling in the range of $\{16, 32, 64, 100\}$. For MGDSPR, the window sizes of long and short-term sequences are ten days and one month, respectively.

For all models, the embedding size is fixed to 64 and $l_h$ is set to be 100. We optimize our method with Adam [14] and use the default learning rate of 0.001 and default mini-batch size of 1024. The number 0f GNN layer $L$ is set to be 2 for the best performance.

*5.1.4 Metrics.* For CTR prediction task, we adopt the commonly used AUC and Log Loss to measure the performance.

## 5.2 Performance Comparision

As shown in Table 2, We have the following observations:

- For CTR baselines, there are similar performances and we use DIEN as an example to illustrate our findings. AMinerGNN and $\text{DIEN}_{\text{f}}$ contribute up to 5.02% AUC and 4.11% AUC promotion compared to $\text{DIEN}_{\text{g}}$ and DIEN, which proves the effectiveness of the query fusion layer. AMinerGNN and $\text{DIEN}_{\text{g}}$ improves over $\text{DIEN}_{\text{f}}$ and DIEN w.r.t. AUC by 26.44% and 23.24%, which proves the effectiveness of GNN embedding layer. Moreover, we find that the improvement brought by the GNN embedding layer is more significant than that by the query fusion layer. The reason is that the GNN embedding layer projects all entities into the same embedding space, which is the prerequisite to accurately modeling the semantic relation between two queries using distance covariance. Another possible reason is that the number of whole keyword behaviors is particularly less than that of user behaviors (in Table 1). This problem may be improved by the increase of Aminer daily UV in the future.
- For product search baselines, all of them underperform than $\text{DIEN}_{\text{g\&f}}$ and $\text{UBR}_{\text{g\&f}}$. The main reason is that they insufficiently recognize the pattern that the user attention towards the input keyword and personalization is non-uniform in different scenarios. A detailed comparison is reported in Section 5.3.
- For two kinds of baselines, modeling temporal interests brings in great improvements, like DIEN compared to DIN, and ALSTP compared to HEM. A clear reason is that the research direction of one user is changing with time.
- Note that we adopt DIEN as our base CTR model rather than UBR. $\text{UBR}_{\text{g\&f}}$ slightly outperforms $\text{DIEN}_{\text{g\&f}}$. However, the online service time of $\text{UBR}_{\text{g\&f}}$ is much bigger than that of $\text{DIEN}_{\text{g\&f}}$, because UBR uses the search engine approach to retrieve the user behaviors, leading to an increase of the online inferring time. We use Locust[5] to simulate 200 simultaneous users and average the response time of CTR-based models[6], which are shown in Table 3. Finally, $\text{DIEN}_{\text{g\&f}}$ is deployed in AMiner as a trade-off between performance and speed.

## 5.3 Scenario test

In this section, we investigate whether AminerGNN adaptively fits for different scenarios. We calulate $\gamma_{uk}$ via equation 3 for each testing batch. $\gamma_{uk} \in [0.5, 1]$, $[0.1, 0.5)$, $[0, 0.1)$ indicate **S1**, **S2**, and **S3**, respectively. The results are shown in Figure 3 and the major findings are as below:

- CTR-based methods perform stably in three scenarios, which proves that two key components of AminerGNN can generalize traditional CTR models to a unified recommender system applied in all scenarios.
- Product search-based models perform better in **S2**, while the performances in the other two scenarios sharply decrease. The reason is the input keyword and personalization are both important in **S2**, which can realize their potentials best. While in the other two scenarios, only one query dominates influence, which is less applicable for product search-based models.

[5]https://locust.io/

[6]We use TorchServe (https://github.com/pytorch/serve) to deploy them.

**Table 2: Overall comparision.**

| | AUC | Log Loss |
|---|---|---|
| $\text{DeepFM}_{\text{base/f/g/g\&f}}$ | 0.635/673/799/837 | 0.674/666/566/517 |
| $\text{DIN}_{\text{base/f/g/g\&f}}$ | 0.657/684/856/876 | 0.645/642/496/465 |
| $\text{DIEN}_{\text{base/f/g/g\&f}}$ | 0.710/711/875/899 | 0.581/577/470/452 |
| $\text{UBR}_{\text{base/f/g/g\&f}}$ | 0.716/719/876/903 | 0.578/571/473/450 |
| HEM | 0.672 | 0.680 |
| ALSTP | 0.840 | 0.512 |
| MGDSPR | 0.871 | 0.478 |
| AMinerGNN(a.k.a. $\text{DIEN}_{\text{g\&f}}$) | **0.899** | **0.452** |

**Table 3: Online response time.**

| | $\text{DIN}_{\text{g\&f}}$ | $\text{DIEN}_{\text{g\&f}}$ | $\text{UBR}_{\text{g\&f}}$ |
|---|---|---|---|
| time(ms) | 97 | 106 | 329 |

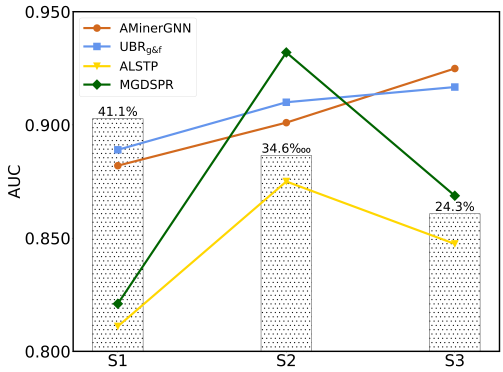


**Figure 3: Scenario test. The background histogram represents the proportion of the number of the testing batch in each scenario.**

**Table 4: Online CTR of compared methods.**

| Metrics | DIEN | MGDSPR | AMinerGNN |
|---|---|---|---|
| UV-CTR | 0.437 | 0.496 | 0.533 |
| PV-CTR | 0.086 | 0.097 | 0.112 |

## 5.4 Online A/B test

We report the online A/B tests of AMinerGNN for a duration of 14 days, from May 17, 2022, to May 30, 2022. About 30 thousand users participated in this A/B test. Two kinds of click-through rate (CTR) on the homepage of AMiner are chosen as metrics. 'UV-CTR' indicates whether a user clicks the recommended paper, irrespective of the number of clicks. 'PV-CTR' is the ratio of clicked papers to all exposed papers. As in Table 4, AMinerGNN achieves 9.6% and 3.7% UV-CTR improvement relative to DIEN and MGDSPR, which demonstrates AMinerGNN works better in paper recommendation with the keyword.

## 6 CONCLUSION

In this paper, we study the recommendation task with two queries. The proposed AMinerGNN can automatically mine user scenario-specific purposes, dynamically recognize the importances of two queries, and attentively fuse two queries to construct a unified and end-to-end recommender system. Compared with the traditional CTR model given one query and product search methods, AMinerGNN performs better and achieves a higher CTR on the AMiner homepage.

# REFERENCES


[1] Qingyao Ai, Yongfeng Zhang, Keping Bi, Xu Chen, and W Bruce Croft. 2017. Learning a hierarchical embedding model for personalized product search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 645–654.

[2] Keping Bi, Qingyao Ai, and W Bruce Croft. 2020. A transformer-based embedding model for personalized product search. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1521–1524.

[3] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2021. Sequential Recommendation with Graph Neural Networks. In *SIGIR*. 378–387.

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

[5] Huizhong Duan and ChengXiang Zhai. 2015. Mining coordinated intent representation for entity search and recommendation. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 333–342.

[6] Huizhong Duan, ChengXiang Zhai, Jinxing Cheng, and Abhishek Gattani. 2013. A probabilistic mixture model for mining and analyzing product search log. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2179–2188.

[7] Huizhong Duan, ChengXiang Zhai, Jinxing Cheng, and Abhishek Gattani. 2013. Supporting keyword search in product database: a probabilistic approach. *Proceedings of the VLDB Endowment* 6, 14 (2013), 1786–1797.

[8] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *WWW*. 2331–2341.

[9] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[10] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, Xiuqiang He, and Zhenhua Dong. 2018. Deepfm: An end-to-end wide & deep learning framework for CTR prediction. *arXiv preprint arXiv:1804.04950* (2018).

[11] Yangyang Guo, Zhiyong Cheng, Liqiang Nie, Yinglong Wang, Jun Ma, and Mohan Kankanhalli. 2019. Attentive long short-term preference modeling for personalized product search. *ACM Transactions on Information Systems (TOIS)* 37, 2 (2019), 1–27.

[12] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM conference on recommender systems*. 43–50.

[13] Shubhra Kanti Karmaker Santu, Parikshit Sondhi, and ChengXiang Zhai. 2017. On application of learning to rank for e-commerce search. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. 475–484.

[14] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[15] Pan Li. 2021. Leveraging Multi-Faceted User Preferences for Improving Click-Through Rate Predictions. In *RecSys*. 864–868.

[16] Sen Li, Fuyu Lv, Taiwei Jin, Guli Lin, Keping Yang, Xiaoyi Zeng, Xiao-Ming Wu, and Qianli Ma. 2021. Embedding-based Product Retrieval in Taobao Search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3181–3189.

[17] Zeyu Li, Wei Cheng, Yang Chen, Haifeng Chen, and Wei Wang. 2020. Interpretable click-through rate prediction through hierarchical attention. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 313–321.

[18] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. 2019. Feature generation by convolutional neural network for click-through rate prediction. In *WWW*. 1119–1129.

[19] Qi Pi, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, Xiaoqiang Zhu, and Kun Gai. 2020. Search-based user interest modeling with lifelong sequential behavior data for click-through rate prediction. In *CIKM*. 2685–2692.

[20] Jiarui Qin, Weinan Zhang, Xin Wu, Jiarui Jin, Yuchen Fang, and Yong Yu. 2020. User behavior retrieval for click-through rate prediction. In *SIGIR*. 2347–2356.

[21] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1149–1154.

[22] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.

[23] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW*. 2022–2032.

[24] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *KDD*. 793–803.

[25] Weinan Zhang, Jiarui Qin, Wei Guo, Ruiming Tang, and Xiuqiang He. 2021. Deep learning for click-through rate estimation. *IJCAI* (2021).

[26] Yuan Zhang, Dong Wang, and Yan Zhang. 2019. Neural IR meets graph embedding: A ranking model for product search. In *The World Wide Web Conference*. 2390–2400.

[27] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *AAAI*, Vol. 33. 5941–5948.

[28] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.