Knowledge Distillation based Contextual Relevance Matching for E-commerce Product Search

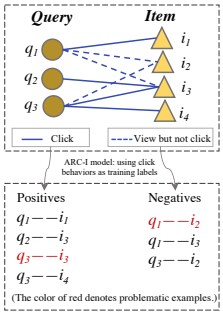
Ziyang Liu[§],Chaokun Wang[§*], Hao Feng[§], Lingfei Wu[†], Liqun Yang[‡]
[§]Tsinghua University,[†]JD.com,[‡]CNAEIT
[§]liu-zy21@mails.thu.edu.cn,[§]chaokun@thu.edu.cn
[§]fh20@mails.thu.edu.cn,[†]lwu@email.wm.edu, [‡]yanglq@cnaeit.com

Abstract

Online relevance matching is an essential task of e-commerce product search to boost the utility of search engines and ensure a smooth user experience. Previous work adopts either classical relevance matching models or Transformer-style models to address it. However, they ignore the inherent bipartite graph structures that are ubiquitous in e-commerce product search logs and are too inefficient to deploy online. In this paper, we design an efficient knowledge distillation framework for e-commerce relevance matching to integrate the respective advantages of Transformer-style models and classical relevance matching models. Especially for the core student model of the framework, we propose a novel method using k-order relevance modeling. The experimental results on large-scale real-world data (the size is 6$\sim$174 million) show that the proposed method significantly improves the prediction accuracy in terms of human relevance judgment. We deploy our method to JD.com online search platform. The A/B testing results show that our method significantly improves most business metrics under price sort mode and default sort mode.

1 Introduction

Relevance matching (Guo et al., 2016; Rao et al., 2019; Wang et al., 2020) is an important task in the field of ad-hoc information retrieval (Zhai and Lafferty, 2017), which aims to return a sequence of information resources related to a user query (Huang et al., 2020; Chang et al., 2021; Sun and Duh, 2020). Generally, texts are the dominant form of user queries and returned information resources. Given two sentences, the target of relevance matching is to estimate their relevance score and then judge whether they are relevant or not. However, text similarity does not mean semantic similarity. For example, while "mac pro 1.7GHz" and "mac lipstick 1.7ml" look alike, they describe two different and irrelevant products. Therefore, relevance matching is important, especially for long-term user satisfaction of e-commerce search (Niu et al., 2020; Xu et al., 2021; Zhu et al., 2020).



	Sentence
q_1	mac pro
q_2	mac lipstick
q_3	mac mini
i_1	Apple MacBook Pro (16-inch, 16GB RAM, 1TB Storage, 2.3GHz Intel Core i9)
i_2	Apple MacBook Pro MLH42LL/A 13.3-inch Laptop
i_3	Little MAC Lipstick 0.06 oz/ 1.77 ml WHIRL
i_4	Apple Mac Mini with Apple M1 Chip (8GB RAM, 256GB SSD Storage)

Figure 1: Shortcoming of the existing relevance matching model. Here we take the ARC-I model as an example. The right part shows the ground truth of queries and item titles. The left part shows two problematic examples in ARC-I, which deviate from the ground truth.

Recently, Transformer-style models (e.g., BERT (Devlin et al., 2019) and ERNIE (Sun et al., 2019b)) have achieved breakthroughs on many NLP tasks and shown satisfactory performance on relevance matching, but they are hard to deploy to the online environment due to their high time complexity. Moreover, these methods cannot deal with the abundant context information (i.e., the neighbor features in a query-item bipartite graph) in e-commerce product search. Last but not least, when applied to real-world scenarios, existing classical relevance matching models directly use user behaviors as labeling information (Figure 1). However, this solution is not directly suitable for relevance matching because user behaviors are often noisy and deviate from relevance signals (Mao et al., 2019; Liu and Mao, 2020).

In this paper, we propose to incorporate bipartite graph embedding into the knowledge distillation framework (Li et al., 2021; Dong et al., 2021; Rashid et al., 2021; Wu et al., 2021b; Zhang et al.,

[*]Chaokun Wang is the corresponding author.

Proceedings of EMNLP 2022 Industry Track, pages 63–76
December 9–11, 2020.

2020) to solve the relevance matching problem in the scene of e-commerce product search. We adopt BERT (Devlin et al., 2019) as the teacher model in this framework. Also, we design a novel model called BERM, **B**ipartite graph **E**mbedding for **R**elevance **M**atching (BERM), which acts as the student model in our knowledge distillation framework. This model captures the 0-order relevance using a word interaction matrix attached with positional encoding and captures the higher-order relevance using the metapath embedding with graph attention scores. For online deployment, it is further distilled into a tiny model BERM-O.

Our main contributions are as follows:

- We formalize the k-order relevance problem in a bipartite graph (Section 2.1) and address it by a knowledge distillation framework with a novel student model called BERM.
- We apply BERM to the e-commerce product search scene with abundant context information (Section 2.4) and evaluate its performance (Section 3). The results indicate that BERM outperforms the state-of-the-art methods.
- To facilitate online applications, we further distill BERM into a faster model, i.e., BERM-O. The results of online A/B testing indicate that BERM-O significantly improves most business metrics under price sort mode and default sort mode.

2 Method

2.1 Problem Definition

We first give the definition of the bipartite graph:

Definition 1 ***Bipartite Graph***. *Given a graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$, it contains two disjoint node sets $\mathcal{U} : \{u_1, u_2, \cdots, u_n\}$ and $\mathcal{V} : \{v_1, v_2, \cdots, v_{n'}\}$. For edge set $\mathcal{E} : \{e_1, \cdots, e_m\}$, each edge e_i connects u_j in $\mathcal{U}$ and v_k in $\mathcal{V}$. In addition, there is a node type mapping function $f_1 : \mathcal{U} \cup \mathcal{V} \to \mathcal{A}$ and an edge type mapping function $f_2 : \mathcal{E} \to \mathcal{R}$. Such a graph $\mathcal{G}$ is called a bipartite graph.*

Example 1 *Given a search log, a query-item bipartite graph is built as shown in Figure 1, where $\mathcal{A}$= $\{Query, Item\}$ and $\mathcal{R}$= $\{Click\}$.*

In a bipartite graph, we use the metapath and metapath instance to incorporate the neighboring node information into relevance matching. They are defined as follows:

Definition 2 ***Metapath and Metapath Instance in Bipartite Graph***. *Given a bipartite graph $\mathcal{G}$=$(\mathcal{U}, \mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$, the metapath P_i=$a_1 \xrightarrow{r_1} a_2 \xrightarrow{r_2} \cdots \xrightarrow{r_l} a_{l+1}$ $(a_j \neq a_{j+1}, 1 \leqslant j \leqslant l)$ is a path from a_1 to a_{l+1} successively through $r_1, r_2, \cdots, r_l$ ($a_j \in \mathcal{A}$, $r_j \in \mathcal{R}$). The length of P_i is denoted as $|P_i|$ and $|P_i|$ =l. For brevity, the set of all metapaths on $\mathcal{G}$ can be represented in regular expression as $P^{\mathcal{G}} = (aa')^+(a|\varepsilon)|(a'a)^+(a'|\varepsilon)$ where $a, a' \in \mathcal{A}$ and $a \neq a'$. The metapath instance p is a definite node sequence instantiated from metapath P_i. All instances of P_i is denoted as $I(P_i)$, then $p \in I(P_i)$.*

Example 2 *As shown in Figure 1, an instance of metapath "Query-Item-Query" is "q_2-i_3-q_3".*

Definition 3 ***k-order Relevance***. *Given a bipartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$, a function $F^k_{rel} : \mathcal{U} \times \mathcal{V} \to [0, 1]$ is called a k-order relevance function on $\mathcal{G}$ if $F^k_{rel}(u_i, v_j) = G(\Phi(u_i), \Phi(v_j)|C_k)$, where $\Phi(\cdot)$ is a function to map each node to a representation vector, $G(.)$ is the score function, $u_i \in \mathcal{U}$, $v_j \in \mathcal{V}$, and context information $C_k = \bigcup_{I_{P_i} \subseteq I(P_i), P_i \in P^{\mathcal{G}}, |P_i|=k} I_{P_i}$.*

Many existing relevance matching models (Huang et al., 2013; Shen et al., 2014; Hu et al., 2014a) ignore context information C_k and only consider the sentences w.r.t. the query and item to be matched, which corresponds to 0-order relevance (for more details, please see the "Related Work" part in Appendix 4). We call it *context-free relevance matching* in this paper. Considering that both the 0-order neighbor (i.e., the node itself) and k-order neighbor ($k \geqslant 1$) are necessary for relevance matching, we argue that a reasonable mechanism should ensure that they can cooperate with each other. Then the research objective of our work is defined as follows:

Definition 4 ***Contextual Relevance Matching***. *Given a bipartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$, the task of contextual relevance matching is to determine the context information C_k on $\mathcal{G}$ and learn the score function $G(\cdot)$.*

2.2 Overview

We propose a complete knowledge distillation framework (Figure 2), whose student model incorporates the context information, for contextual relevance matching in e-commerce product search. The main components of this framework are described as follows:

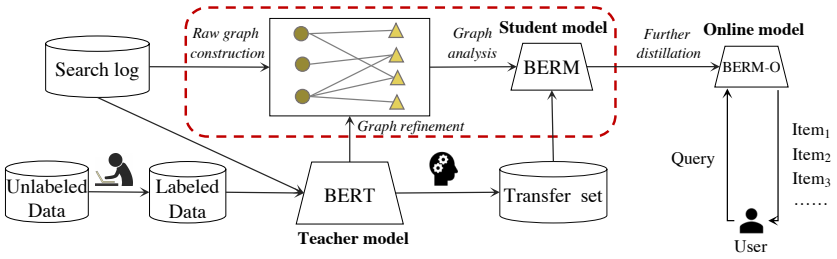


Figure 2: The e-commerce knowledge distillation framework proposed in our work. Three models are used in this framework: teacher model BERT, student model BERM, and online model BERM-O.

- **Graph construction.** We first construct a raw bipartite graph $\mathcal{G}$ based on the search data collected from JD.com. Then we construct a knowledge-enhanced bipartite graph $\mathcal{G}'$ with the help of BERT, which is fine-tuned by the human-labeled relevance data.

- **Student model design.** We design a novel student model BERM corresponding to the score function $G(\cdot)$ in Definition 4. Specifically, macro and micro matching embeddings are derived in BERM to capture the sentence-level and word-level relevance matching signal, respectively. Also, based on the metapaths "Q-I-Q" and "I-Q-I", we design a node-level encoder and a metapath-instance-level aggregator to derive metapath embeddings.

- **Online application.** To serve online search, we conduct further distillation to BERM and obtain BERM-O, which is easy to be deployed online.

2.3 Bipartite Graph Construction

We introduce the external knowledge from BERT to refine the raw user behavior graph $\mathcal{G}$ into a knowledge-enhanced bipartite graph $\mathcal{G}'$. The whole graph construction includes the following phases.

Fine-tuning BERT. We use the BERT model as the teacher model in our framework. BERT is pre-trained on a large text corpus and fine-tuned on our in-house data where the positive examples and negative examples are human-labeled and cover various item categories. The fine-tuned BERT is equipped with good relevance discrimination and thus acts as an expert in filtering noisy data. For each example pair p_i in the transfer set S_{transfer}, we use BERT to predict its score y_i as the training label of the student model BERM.

Behavior graph construction. The user behavior graph $\mathcal{G}$ is built on the user search log over six months which records click behaviors and purchase behaviors as well as their frequencies. Each edge in $\mathcal{G}$ represents an existing click behavior or purchase behavior between the given query and item.

Knowledge-enhanced graph refinement. The click behavior edges are dense and highly noisy, so we leverage the fine-tuned BERT model to refine $\mathcal{G}$. Specifically, we retain all the raw purchase behavior edges, and meanwhile use the knowledge generated by the fine-tuned BERT to refine the click behavior edges. We set two thresholds α and β to determine which raw edges are removed and which new edges are added. This strategy helps remove the noise in user behaviors, and at the same time retrieve the missing but relevant neighbors which cannot be captured by user behaviors. To preserve important neighbors, for each anchor node, we rank its 1-hop neighbors with the priority of "purchase>high click>low click" and select the top two of them as the final neighbor list, i.e., the neighbor list of a query node Q is represented as $[I_{\text{top1}}, I_{\text{top2}}]$ and the neighbor list of a query node I is represented as $[Q_{\text{top1}}, Q_{\text{top2}}]$. The algorithm of graph construction is provided in Appendix A.

2.4 BERM Model

In this part, we describe BERM in detail, including 0-order relevance modeling, k-order relevance modeling, and overall learning objective.

2.4.1 0-order Relevance Modeling

The whole structure of BERM includes both the 0-order relevance modeling and k-order relevance modeling. This subsection introduces the 0-order relevance modeling which captures sentence-level and word-level matching signals by incorporating the macro matching embedding and micro matching embedding, respectively.

Macro and micro matching embeddings. Each word is represented by a d-dimensional embedding vector, which is trained by Word2Vec (Mikolov et al., 2013). The i-th word's embedding of query

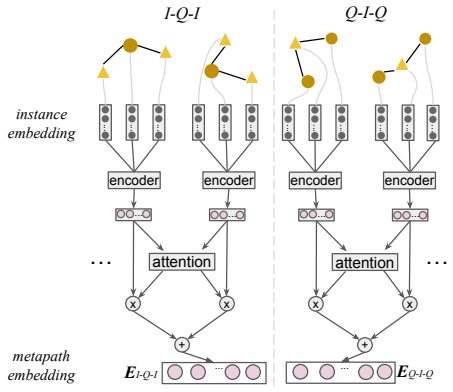


Figure 3: Calculation process of metapath embeddings.

Q (or item title I) is denoted as $\boldsymbol{E}_Q^i \in \mathbb{R}^d$ (or $\boldsymbol{E}_I^i \in \mathbb{R}^d$). To capture sentence-level and word-level matching signals, we employ macro matching embedding and micro matching embedding, respectively. For the macro matching embedding, taking query Q with l_Q words and item I with l_I words as examples, their macro embeddings $\boldsymbol{E}_{\text{seq}}^Q, \boldsymbol{E}_{\text{seq}}^I \in \mathbb{R}^d$ are calculated by the column-wise mean value of $\boldsymbol{E}_Q \in \mathbb{R}^{l_Q \times d}, \boldsymbol{E}_I \in \mathbb{R}^{l_I \times d}$:

$$\boldsymbol{E}_{\text{seq}}^Q = \frac{1}{l_Q} \sum_{i=1}^{l_Q} \boldsymbol{E}_Q^i, \quad \boldsymbol{E}_{\text{seq}}^I = \frac{1}{l_I} \sum_{i=1}^{l_I} \boldsymbol{E}_I^i . \tag{1}$$

For the micro matching embedding, we first build an interaction matrix $\boldsymbol{M}_{\text{int}} \in \mathbb{R}^{l_Q \times l_I}$ whose (i,j)-th entry is the dot product of $\boldsymbol{E}_Q^i$ and $\boldsymbol{E}_I^j$:

$$\boldsymbol{M}_{\text{int}} = \left\{ m_{\text{int}}^{i,j} \right\}_{l_Q \times l_I}, \quad m_{\text{int}}^{i,j} = \langle \boldsymbol{E}_Q^i, \boldsymbol{E}_I^j \rangle . \tag{2}$$

Then the micro matching embedding $\boldsymbol{E}_{\text{int}} \in \mathbb{R}^{l_Q l_I}$ is the vectorization of $\boldsymbol{M}_{\text{int}}$, i.e., $\boldsymbol{E}_{\text{int}} = \text{vec}(\boldsymbol{M}_{\text{int}})$.

2.4.2 k-order Relevance Modeling

The k-order relevance model contains a node-level encoder and a metapath-instance-level aggregator.

Node-level encoder. The input of the node-level encoder is node embeddings and its output is an instance embedding (i.e., the embedding of a metapath instance). Specifically, to obtain the instance embedding, we integrate the embeddings of neighboring nodes into the anchor node embedding with a mean encoder. Taking "Q-I_{top1}-Q_{top1}" as an example, we calculate its embedding $\boldsymbol{E}_{Q\text{-}I_{\text{top1}}\text{-}Q_{\text{top1}}} \in \mathbb{R}^d$ as follows:

$$\boldsymbol{E}_{Q\text{-}I_{\text{top1}}\text{-}Q_{\text{top1}}} = \text{MEAN}(\boldsymbol{E}_{\text{seq}}^Q, \boldsymbol{E}_{\text{seq}}^{I_{\text{top1}}}, \boldsymbol{E}_{\text{seq}}^{Q_{\text{top1}}}). \tag{3}$$

The metapath instance bridges the communication gap between different types of nodes and can be used to update the anchor node embedding from structure information.

Metapath-instance-level aggregator. The inputs of the metapath-instance-level aggregator are instance embeddings and its output is a metapath embedding. Different metapath instances convey different information, so they have various effects on the final metapath embedding. However, the mapping relationship between the instance embedding and metapath embedding is unknown. To learn their relationship automatically, we introduce the "graph attention" mechanism to generate metapath embeddings (Wu et al., 2021a; Liu et al., 2022). Taking metapath "Q-I-Q" as an example, we use graph attention to represent the mapping relationship between "Q-I-Q" and its instances. The final metapath embedding $\boldsymbol{E}_{Q\text{-}I\text{-}Q} \in \mathbb{R}^d$ is calculated ($\boldsymbol{E}_{I\text{-}Q\text{-}I} \in \mathbb{R}^d$ is calculated similarly) by accumulating all instance embeddings with attention scores $\text{Att}_1, \text{Att}_2, \text{Att}_3, \text{Att}_4 \in \mathbb{R}^+$:

$$\begin{aligned} \boldsymbol{E}_{Q\text{-}I\text{-}Q} = \sigma(&\text{Att}_1 \cdot \boldsymbol{E}_{Q\text{-}I_{\text{top1}}\text{-}Q_{\text{top1}}} + \text{Att}_2 \cdot \boldsymbol{E}_{Q\text{-}I_{\text{top1}}\text{-}Q_{\text{top2}}} \\ &+ \text{Att}_3 \cdot \boldsymbol{E}_{Q\text{-}I_{\text{top2}}\text{-}Q_{\text{top1}}} + \text{Att}_4 \cdot \boldsymbol{E}_{Q\text{-}I_{\text{top2}}\text{-}Q_{\text{top2}}}), \end{aligned} \tag{4}$$

where $\sigma(\cdot)$ is the activation function of LeakyReLU. Though Att_i can be set as a fixed value, we adopt a more flexible way, i.e., using the neural network to learn Att_i automatically. Specifically, we feed the concatenation of the anchor node embedding and metapath instance embedding into a one-layer neural network (its weight is $\boldsymbol{W}_{\text{att}} \in \mathbb{R}^{6d \times 4}$ and its bias is $\boldsymbol{b}_{\text{att}} \in \mathbb{R}^{1 \times 4}$) with a softmax layer, which outputs an attention distribution:

$$(\text{Att}_i)_{1 \le i \le 4} = \text{softmax}(\boldsymbol{E}_{\text{concat}} * \boldsymbol{W}_{\text{att}} + \boldsymbol{b}_{\text{att}}), \tag{5}$$

$$\begin{aligned} \boldsymbol{E}_{\text{concat}} = [&\boldsymbol{E}_{\text{seq}}^Q | \boldsymbol{E}_{\text{seq}}^I | \boldsymbol{E}_{Q\text{-}I_{\text{top1}}\text{-}Q_{\text{top1}}} | \boldsymbol{E}_{Q\text{-}I_{\text{top1}}\text{-}Q_{\text{top2}}} \\ &| \boldsymbol{E}_{Q\text{-}I_{\text{top2}}\text{-}Q_{\text{top1}}} | \boldsymbol{E}_{Q\text{-}I_{\text{top2}}\text{-}Q_{\text{top2}}}]. \end{aligned} \tag{6}$$

The above process is shown in Figure 3.

Embedding fusion. By the 0-order and k-order relevance modeling, three types of embeddings are generated, including macro matching embedding ($\boldsymbol{E}_{\text{seq}}^Q, \boldsymbol{E}_{\text{seq}}^I \in \mathbb{R}^d$), micro matching embedding ($\boldsymbol{E}_{\text{int}} \in \mathbb{R}^{l_Q l_I}$), and metapath embedding ($\boldsymbol{E}_{Q\text{-}I\text{-}Q}, \boldsymbol{E}_{I\text{-}Q\text{-}I} \in \mathbb{R}^d$). We concatenate them together and feed the result to a three-layer neural network (its weights are $\boldsymbol{W}_0 \in \mathbb{R}^{(4d+l_Q l_I) \times d}, \boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{d \times d}, \boldsymbol{W}_3 \in \mathbb{R}^{d \times 1}$ and biases are $\boldsymbol{b}_0, \boldsymbol{b}_1, \boldsymbol{b}_2 \in \mathbb{R}^{1 \times d}, \boldsymbol{b}_3 \in \mathbb{R}^{1 \times 1}$), which outputs the final relevance estimation score $\hat{y}_i$:

$$\hat{y}_i = \text{Sigmoid}(\boldsymbol{E}_3 * \boldsymbol{W}_3 + \boldsymbol{b}_3), \tag{7}$$

$$\boldsymbol{E}_{j+1} = \text{ReLU}(\boldsymbol{E}_j * \boldsymbol{W}_j + \boldsymbol{b}_j), \boldsymbol{E}_0 = \boldsymbol{E}_{all}, j = 0, 1, 2, \tag{8}$$

$$\boldsymbol{E}_{all} = [\boldsymbol{E}_{\text{seq}}^Q | \boldsymbol{E}_{\text{seq}}^I | \boldsymbol{E}_{\text{int}} | \boldsymbol{E}_{Q\text{-}I\text{-}Q} | \boldsymbol{E}_{I\text{-}Q\text{-}I}]. \tag{9}$$

2.4.3 Overall Learning Objective

We evaluate the cross-entropy error on the estimation score $\hat{y}_i$ and label y_i (note that $y_i \in [0, 1]$ is the score of the teacher model BERT), and then minimize the following loss function:

$$\mathcal{L} = -\sum_{i=1}^{\tilde{n}} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i), \qquad (10)$$

where $\tilde{n}$ is the number of examples. We also analyze the complexities of BERT, BERM, and BERM-O in Appendix C.

3 Experiments

In this section, we present the offline and online experimental results of BERM [*].

3.1 Experimental Setting

Datasets. We collect three datasets from the search platform of JD.com, including the "Electronics" category (Data-E), all-category (Data-A), and sampled all-category (Data-S). In the platform, there are mainly three different levels of item categories: Cid_1 (highest level, e.g., "Electronics"), Cid_2 (e.g., "Mobile phone"), and Cid_3 (lowest level, e.g., "5G phone"). Data-A, Data-S, and Data-E have different data distributions. Specifically, Data-A covers all first-level categories Cid_1 in the platform; Data-S is generated by uniformly sampling 5,000 items from Cid_1; Data-E only focuses on the category of "Electronics" in Cid_1. Details of Data-E, Data-A, and Data-S are reported in Table 1.

For the training data S_{train} (also called S_{transfer}), the collected user behaviors include click and purchase. For the testing data S_{test}, whose queries are disjointed with those of S_{train}, we use human labeling to distinguish between relevant and irrelevant items. Specifically, editors are asked to assess the relevance scores between queries and items. In JD.com platform, the candidate set of relevance scores is $\{1, 2, 3, 4, 5\}$, where 5 means most relevant and 1 means least relevant. To simplify it, we use binary labeling including the positive label (i.e., 4 or 5) and negative label (i.e., 1, 2, or 3).

Evaluation Metrics. To measure the performance of baseline methods and our BERM, we use three kinds of evaluation metrics, including Area Under the receiver operating characteristic Curve (AUC), F1-score, and False Negative Rate (FNR).

[*]We provide the description of baselines, implementation details, and additional experiments in Appendix D.1, D.2, E (Code URL: https://github.com/Young0222/EMNLP-BERM).

Table 1: Statistics of the used datasets.

Set	Name	Data-E	Data-A	Data-S
S_{train}	# Example	6,369,396	174,863,375	11,397,439
	# $\text{Node}_{\text{query}}$	398,824	5,952,020	3,284,480
	# $\text{Node}_{\text{item}}$	728,405	49,517,217	1,307,557
	# Edge	5,070,460	159,205,320	7,525,355
	# Click	1,471,079,596	5,109,731,591	1,431,899,847
	# Purchase	33,285,887	322,151,488	118,495,170
S_{test}	# Example	30,563	39,743	39,743
	# $\text{Node}_{\text{query}}$	3,374	3,108	3,108
	# $\text{Node}_{\text{item}}$	16,137	30,097	30,097
	# Edge	18,988	30,661	30,661

Table 2: Comparisons on Data-E and Data-S. In each column, the best result is bolded and the runner-up is underlined. The symbol of "↓" represents that the lower value corresponds to better performance. "I, II, III" represent the representation-focused, interaction-focused, and both-focused relevance matching models, respectively. "IV" represents the graph neural network models.

	Model	Data-E			Data-S		
		AUC	F1-score	FNR(↓)	AUC	F1-score	FNR(↓)
I	DSSM	0.6246	0.6923	0.9953	0.8219	0.8691	1.0000
	MVLSTM	0.8602	0.8055	0.3416	0.7877	0.8857	0.7802
	ARC-I	0.8343	0.7949	0.3857	0.6919	0.8750	0.9388
II	DRMM	0.6720	0.6891	0.7692	0.6781	0.8722	0.9401
	MatchPyramid	0.7826	0.7481	0.5615	0.7859	0.8786	0.8475
	ARC-II	0.8128	0.7864	0.4377	0.7606	0.8784	0.9076
	K-NRM	0.7462	0.7291	0.6510	0.7314	0.8733	0.9081
	DRMM-TKS	0.7678	0.7383	0.5462	0.7793	0.8789	0.7893
	Conv-KNRM	0.8369	0.7879	0.3469	0.8029	0.8789	0.8913
	ESIM	0.8056	0.7769	0.3373	0.7987	0.8623	1.0000
III	Duet	0.7693	0.7219	0.8173	0.7968	0.8754	0.9458
	BERT2DNN	0.8595	0.8037	0.3464	0.8313	0.9061	0.4450
IV	GAT	0.7526	0.7361	0.7529	0.7411	0.8746	0.9234
	GraphSAGE-Mean	0.7493	0.7330	0.7422	0.7406	0.8719	0.9119
	GraphSAGE-LSTM	0.7588	0.7509	0.6536	0.7529	0.8743	0.8652
	TextGNN	0.8310	0.8029	0.4525	0.8277	0.8779	0.7549
	GEPS	0.8405	0.8037	0.4892	0.8254	0.8794	0.6340
	BERM (ours)	**0.8785**	**0.8256**	**0.2966**	**0.8758**	**0.9079**	**0.3625**

The low value of FNR indicates the low probability of fetching irrelevant items, which is closely related to the user's search experience. Therefore, we include it in the evaluation metrics.

3.2 Offline Performance

We compare BERM with 12 state-of-the-art relevance matching methods and 5 graph neural network models on our in-house product search data. The results are shown in Table 2. Because some baseline methods (e.g., DRMM and ESIM) have high time complexities, we use Data-E and Data-S for training and testing models.

As shown in Table 2, BERM outperforms all the baselines according to the metrics of AUC, F1-score, and FNR. More specifically, we have the following findings: 1) Compared to the second-

Table 3: Cases of e-commerce product search. "y_i" is the prediction score of the teacher model BERT and "$\hat{y}_i$" is the relevance estimation score of the student model BERM.

Query	Item title	Human labeling	y_i	$\hat{y}_i$
whistle	Li Ning whistle for basketball or volleyball game	Positive	0.9961	0.9681
women's dance shoe	Sansha modern dance shoe P22LS (black, women)	Positive	0.9973	0.9908
violin adult	FineLegend 1/8 violin FLV1114	Positive	0.9950	0.9769
skating knee panels	RMT sports knee panels (black, L size)	Positive	0.9652	0.9841
DJI g1200a	DJI Mavic Mini unmanned aerial vehicle	Negative	0.0049	0.0514
Berkshire Hathaway Letters to Shareholders	The Snowball: Warren Buffett and the Business of Life	Negative	0.0258	0.0874
My brother called Shun Liu, ZHU SU JIN	Brothers: A Novel; Author: Yu Hua	Negative	0.1624	0.0263
nissan thermos cup	Disney thermos cup 500ML	Negative	0.4938	0.2513
java web exercises	JSP project development Case Full Record	Negative	0.5106	0.3111

best method MVLSTM (BERT2DNN), BERM surpasses it 1.83% (4.45%) according to AUC on Data-E (Data-S). Furthermore, BERM achieves the lowest value of FNR on both Data-E and Data-S. This implies that BERM can easily identify irrelevant items so that it can return a list of satisfactory items in the real-world scene. 2) The collected training data have imbalanced classes (i.e., the positive examples are far more than the negative examples), which poses a challenge to model learning. Most baselines are sensitive to class imbalance. Since BERM learns explicit node semantics by integrating the neighboring node information, our method is robust when the data are imbalanced.

3.3 Case Study

Apart from the above quantitative analysis, we conduct qualitative analysis based on some cases of e-commerce product search. For these cases, we list the query phrase, item title, human labeling, score of BERT, and score of BERM in Table 3. We have the following empirical conclusions: 1) Most of the student's scores are close to the teacher's, which indicates the success of the proposed knowledge distillation framework. 2) Some cases imply that context information is necessary for relevance matching. For example, for the query "nissan thermos cup", the teacher model cannot explicitly judge whether or not the item entitled "Disney thermos cup 500ML" is relevant to it. With the help of context information in the query-item bipartite graph, BERM can recognize that this query is related to "nissan", rather than "Disney".

3.4 Deployment & Online A/B Testing

We conduct further distillation to BERM and obtain a lighter model BERM-O whose basic structure is a two-layer neural network. The process of further distillation is almost the same as the first knowledge distillation. The transfer set generated by further distillation has graph-context labels. To further evaluate BERM-O's performance in the real search scene, we deploy it to JD.com online search platform. On this platform, there are about one hundred million daily active users (DAU) and two billion items. It processes over 150 million search queries per day. The online baseline group BERT2DNN (Jiang et al., 2020) and control group BERM-O are deployed in a cluster, where each node is with 64 core Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.10GHz, 256GB RAM as well as 4 NVIDIA TESLA P40 GPU cards. For both groups, the only needed input data are queries and item titles, which can be easily caught from the online environment. Since BERM-O is lighter than BERT or BERM, deploying it to the online search chain requires less engineering work in the system.

Table 4: Online performance of BERM-O under price sort mode and default sort mode.

Metric	Price sort mode		Default sort mode	
	Improvement	P-value	Improvement	P-value
UV-value	5.713%	3.20e-2	0.5013%	1.10e-1
UCVR	1.540%	7.81e-2	0.3058%	1.75e-2
CVR	1.829%	1.01e-2	0.1218%	1.60e-1
RPM	5.587%	3.03e-2	0.6886%	2.32e-2

Online results. We compare BERM-O with BERT2DNN (Jiang et al., 2020) which is our online baseline model using knowledge distillation without context information. The results of A/B testing are reported in Table 4. These results are from one observation lasting more than ten days. Four widely-used online business metrics are adopted 1) conversion rate (CVR): the average order number of each click behavior, 2) user conversion rate (UCVR): the average order number of each user, 3) unique visitor value (UV-value): the average gross merchandise volume of each user, and 4) revenue per mile (RPM): the average gross merchandise volume of each retrieval behavior. The results show

that BERM-O outperforms BERT2DNN in the platform according to all of the business metrics. For example, BERM-O significantly improves 5.7% (relative value) of UV-value under price sort mode.

4 Related Work

4.1 Classical Relevance Matching Models

The classical relevance matching models use the deep learning technique to learn vector representations containing the semantics of words or sequences. The prevailing methods are either representation-focused (e.g., DSSM (Huang et al., 2013), CDSSM (Shen et al., 2014), and ARC-I (Hu et al., 2014b)) or interaction-focused (e.g., MatchPyramid (Pang et al., 2016a), ARC-II (Hu et al., 2014b), and ESIM (Chen et al., 2017)). The representation-focused methods learn the low-dimensional representations of both sentences and then predict their relationship by calculating the similarity between the two representations. The interaction-focused methods learn an interaction representation of both sentences based on the calculation from word-level to sentence-level.

However, the above methods ignore the inherent context information contained in search logs (Qin et al., 2022; Roßrucker, 2022). In this work, we incorporate the advantages of representation-based and interaction-based embeddings into BERM, which is focused on contextual relevance matching.

4.2 Transformer-style Models

More recently, Transformer-based models (Chen et al., 2021; Chi et al., 2021; Lin et al., 2021; Reid et al., 2021) have achieved breakthroughs on many NLP tasks and reached human-level accuracy. The representative models include BERT (Devlin et al., 2019), ERNIE (Sun et al., 2019b), and RoBERTa (Liu et al., 2019b). Additionally, GRMM (Zhang et al., 2021) and GHRM (Yu et al., 2021) use graph information to enforce the relevance matching model for information retrieval.

However, the multi-layer stacked Transformer structure in these models leads to high time complexity, so they are hard to deploy online. In this work, we use BERT to generate the supervised information of BERM and refine the noisy behavior data. Also, GRMM and GHRM are essentially different from ours in the definition of graphs. In their constructed graph, nodes are unique words and edges are the co-occurrent relationships. In this work, we leverage query phrases (or item titles) as nodes and user behaviors as edges, which is more suitable for the product search problem.

4.3 Online Knowledge Distillation Methods

Knowledge distillation is firstly proposed in (Hinton et al., 2015). Its main idea is to transfer the knowledge generated by a massive teacher model into a light student model. Because of the low complexity of the student model, it is easy to deploy the student model to the online platform. Considering the strong semantic understanding ability of BERT, some studies exploit the potential of BERT as the teacher model of knowledge distillation. Two types of design principles are general: isomorphic principle and isomeric principle. Specifically, the distillation methods that follow the isomorphic principle use the same model architecture for teacher and student models, such as TinyBERT (Jiao et al., 2020), BERT-PKD (Sun et al., 2019a), MTDNN (Liu et al., 2019a), and DistilBERT (Sanh et al., 2019). As a more advanced design principle, the isomeric principle uses different model architectures for teacher and student models, such as Distilled BiLSTM (Tang et al., 2019) and BERT2DNN (Jiang et al., 2020).

Although the above methods reduce the total time costs by learning a light student model, they ignore the context information in the real search scene. Our proposed knowledge distillation framework follows the isomeric principle and further integrates context information into the student model by bipartite graph embedding.

5 Conclusions and Future Work

In this paper, we propose the new problem of contextual relevance matching in e-commerce product search. Different from the previous work only using the 0-order relevance modeling, we propose a novel method of the k-order relevance modeling, i.e., employing bipartite graph embedding to exploit the potential context information in the query-item bipartite graph. Compared to the state-of-the-art relevance matching methods, the new method BERM performs robustly in the experiments. We further distill BERM into BERM-O and deploy BERM-O to JD.com online e-commerce product search platform. The results of A/B testing indicate that BERM-O improves the user's search experience significantly. In the future, we plan to apply our method to other e-commerce applications such as recommender systems and advertisements.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China (No. 61872207) and JD.com, Inc. Chaokun Wang is the corresponding author.

References

Wei-Cheng Chang, Daniel Jiang, Hsiang-Fu Yu, Choon Hui Teo, Jiong Zhang, Kai Zhong, Kedarnath Kolluri, Qie Hu, Nikhil Shandilya, Vyacheslav Ievgrafov, et al. 2021. Extreme multi-label learning for semantic matching in product search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2643–2651.

Pu-Chin Chen, Henry Tsai, Srinadh Bhojanapalli, Hyung Won Chung, Yin-Wen Chang, and Chun-Sung Ferng. 2021. A simple and effective positional encoding for transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2974–2988, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668.

Zewen Chi, Li Dong, Shuming Ma, Shaohan Huang, Saksham Singhal, Xian-Ling Mao, Heyan Huang, Xia Song, and Furu Wei. 2021. mT6: Multilingual pretrained text-to-text transformer with translation pairs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1671–1683, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186.

Chenhe Dong, Yaliang Li, Ying Shen, and Minghui Qiu. 2021. HRKD: Hierarchical relational knowledge distillation for cross-domain language model compression. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3126–3136, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014a. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2042–2050.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014b. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050.

Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2553–2561.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338.

Yunjiang Jiang, Yue Shang, Ziyang Liu, Hongwei Shen, Yun Xiao, Wei Xiong, Sulong Xu, Weipeng Yan, and Di Jin. 2020. Bert2dnn: Bert distillation with massive unlabeled data for online e-commerce search. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 212–221. IEEE.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4163–4174.

Lei Li, Yankai Lin, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. 2021. Dynamic knowledge distillation for pre-trained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ye Lin, Yanyang Li, Tong Xiao, and Jingbo Zhu. 2021. Bag of tricks for optimizing transformer efficiency. In *EMNLP*.

Hao Liu, Jindong Han, Yanjie Fu, Yanyan Li, Kai Chen, and Hui Xiong. 2022. Unified route representation learning for multi-modal transportation recommendation with spatiotemporal pre-training. *The VLDB Journal*, pages 1–18.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv preprint arXiv:1904.09482*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yiqun Liu and Jiaxin Mao. 2020. " revisiting information retrieval tasks with user behavior models" by yiqun liu and jiaxin mao with martin vesely as coordinator. *ACM SIGWEB Newsletter*, pages 1–8.

Jiaxin Mao, Zhumin Chu, Yiqun Liu, Min Zhang, and Shaoping Ma. 2019. Investigating the reliability of click models. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 125–128.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 1291–1299. ACM.

Xichuan Niu, Bofang Li, Chenliang Li, Rong Xiao, Haochuan Sun, Hongbo Deng, and Zhenzhong Chen. 2020. A dual heterogeneous graph attention network to improve long-tail performance for shop search in e-commerce. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3405–3415.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016a. Text matching as image recognition. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016b. Text matching as image recognition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2793–2799.

Chuan Qin, Kaichun Yao, Hengshu Zhu, Tong Xu, Dazhong Shen, Enhong Chen, and Hui Xiong. 2022. Towards automatic job description generation with capability-aware neural networks. *IEEE Transactions on Knowledge and Data Engineering*.

Jinfeng Rao, Linqing Liu, Yi Tay, Wei Yang, Peng Shi, and Jimmy Lin. 2019. Bridging the gap between relevance matching and semantic matching for short text similarity modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5370–5381.

Ahmad Rashid, Vasileios Lioutas, Abbas Ghaddar, and Mehdi Rezagholizadeh. 2021. Towards zero-shot knowledge distillation for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6551–6561, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Machel Reid, Edison Marrese-Taylor, and Yutaka Matsuo. 2021. Subformer: Exploring weight sharing for parameter efficiency in generative transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4081–4090, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Georg P Roßrucker. 2022. State-of-the-art survey on web search. In *The Autonomous Web*, pages 1–24. Springer.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd international conference on world wide web*, pages 373–374.

Shuo Sun and Kevin Duh. 2020. CLIRMatrix: A massively large collection of bilingual and multilingual datasets for cross-lingual information retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4160–4170, Online. Association for Computational Linguistics.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019a. Patient knowledge distillation for bert model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4323–4332.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019b. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2835–2841.

Junmei Wang, Min Pan, Tingting He, Xiang Huang, Xueyan Wang, and Xinhui Tu. 2020. A pseudo-relevance feedback framework combining relevance matching and semantic matching for information retrieval. *Information Processing & Management*, 57(6):102342.

Xinle Wu, Dalin Zhang, Chenjuan Guo, Chaoyang He, Bin Yang, and Christian S Jensen. 2021a. Autocts: Automated correlated time series forecasting. *Proceedings of the VLDB Endowment*, 15(4):971–983.

Yimeng Wu, Mehdi Rezagholizadeh, Abbas Ghaddar, Md Akmal Haidar, and Ali Ghodsi. 2021b. Universal-KD: Attention-based output-grounded intermediate layer knowledge distillation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7649–7661, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 55–64. ACM.

Song Xu, Haoran Li, Peng Yuan, Yujia Wang, Youzheng Wu, Xiaodong He, Ying Liu, and Bowen Zhou. 2021. K-PLUG: Knowledge-injected pre-trained language model for natural language understanding and generation in E-commerce. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1–17, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xueli Yu, Weizhi Xu, Zeyu Cui, Shu Wu, and Liang Wang. 2021. Graph-based hierarchical relevance matching signals for ad-hoc retrieval. *CoRR*, abs/2102.11127.

Chengxiang Zhai and John Lafferty. 2017. A study of smoothing methods for language models applied to ad hoc information retrieval. In *ACM SIGIR Forum*, volume 51, pages 268–276. ACM New York, NY, USA.

Yuan Zhang, Dong Wang, and Yan Zhang. 2019. Neural ir meets graph embedding: A ranking model for product search. *The World Wide Web Conference*.

Yuan Zhang, Xiaoran Xu, Hanning Zhou, and Yan Zhang. 2020. Distilling structured knowledge into embeddings for explainable and accurate recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 735–743.

Yufeng Zhang, Jinghao Zhang, Zeyu Cui, Shu Wu, and Liang Wang. 2021. A graph-based relevance matching model for ad-hoc retrieval. *CoRR*, abs/2101.11873.

Jason Zhu, Yanling Cui, Yuming Liu, Hao Sun, Xue Li, Markus Pelger, Tianqi Yang, Liangjie Zhang, Ruofei Zhang, and Huasha Zhao. 2021. Textgnn: Improving text encoder via graph neural network in sponsored search. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 2848–2857. ACM / IW3C2.

Tiangang Zhu, Yue Wang, Haoran Li, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2020. Multimodal joint attribute prediction and value extraction for E-commerce product. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2129–2139, Online. Association for Computational Linguistics.

A Graph Construction Algorithm

We provide the complete algorithm of graph construction in Algorithm 1.

Algorithm 1 Bipartite Graph Construction

Input: Thresholds α and β; Collected dataset $S_{\text{input}}=\{p_i\}_{\tilde{n}}$ (note that $\tilde{n}$ is the number of examples); Raw user behavior graph $\mathcal{G}=(\mathcal{U},\mathcal{V},\mathcal{E},\mathcal{A},\mathcal{R})$ where $\mathcal{E}=\{e_1,\cdots,e_m\}$, $\mathcal{A}=\{Query, Item\}$, $\mathcal{R}=\{Click, Purchase\}$.

Output: Transfer set $S_{\text{transfer}}=\{p_i;y_i\}_{\tilde{n}}$ where y_i is the training label of query-item pair p_i ($y_i\in[0,1]$); Refined bipartite graph $\mathcal{G}'=(\mathcal{U},\mathcal{V},\mathcal{E}',\mathcal{A},\mathcal{R})$.

```
Initialize E': E'=E.
Fine-tune BERT on the human-labeled data.
Use the fine-tuned BERT to predict on S_input and then obtain S_transfer={p_i; y_i}_ñ.
for p_i, y_i in S_transfer do
    Build an edge between the pair p_i and denote it as e_i=edge(p_i).
    if e_i∈E then
        if f_2(e_i)=Purchase then
            continue;
        else if y_i < α then
            E'=E' \ {e_i}
        end if
    else if y_i > β then
        E'=E' ∪ {e_i}
    end if
end for
```

B Word Embedding in E-commerce Scene

In this part, we introduce the details of word embedding generation used in this work. In e-commerce scene, the basic representation of a query or an item is an intractable problem. On one hand, it is infeasible to represent queries and items as individual embeddings due to the unbounded entity space. On the other hand, product type names (like "iphone11") or attribute names (like "256GB") have special background information and could contain complex lexicons such as different languages and numerals. To address these problems, we adopt word embedding in BERM, which dramatically reduces the representation space. Also, we treat contiguous numerals, contiguous English letters, or single Chinese characters as one word and only retain the high-frequency words (such as the words occurring more than fifty times in a six-month search log) in the vocabulary. The final vocabulary is only in the tens of thousands, which saves memory consumption and lookup time of indexes by a large margin.

C Complexity Analysis

In this section, we analyze the time and space complexities of BERT (teacher model), BERM (student model), and BERM-O (online model).

C.1 Time Complexity

For the lookup operation on the static vocabulary table (i.e., a word embedding table whose size is n_w), the time complexities of BERT, BERM, and BERM-O are the same, i.e., O(n_w). For the model calculation part, BERT uses Transformer networks. We denote the word embedding size, head number, network number, query length, and item length as d_1, h_1, k_1, l_Q, l_I, respectively. For the one-layer multi-head attention mechanism, the complexity of linear mapping (input part) is O($\frac{1}{h_1}(l_Q+l_I)d_1^2$), the complexity of attention operation is O($h_1 l_Q^2 d + h l_I^2 d_1$), and the complexity of linear mapping (output part) is O($(l_Q+l_I)d_1^2$). Therefore, the total model calculation complexity of k_1-layer BERT is O($\frac{h_1+1}{h_1}(l_Q+l_I)d_1^2 k_1 + (l_Q^2+l_I^2)h_1 d_1 k_1$). For the student model BERM, we denote the word embedding size, hidden size, and network number as d, h_2, k_2, respectively. The complexity of calculating micro matching embedding is O($l_Q l_I d$), which is far more than that of calculating micro matching embedding. The complexity of k_2-layer DNN is O($h_2 d k_2$). Therefore, the total model calculation complexity of k_2-layer BERM is O($l_Q l_I d + h_2 d k_2$). For the online model BERM-O, we denote the word embedding size, hidden size, and network number as d_3, h_3, k_3, respectively. The model calculation complexity of k_3-layer BERM-O is O($d_3 h_3 k_3$). Note that the complexity of BERM-O is independent of l_Q and l_I because BERM-O only receives sentence embeddings and does not calculate word-level matching signals. Based on the above analysis, we can conclude that BERM is more efficient than BERT and meanwhile BERM-O has more advantages than BERM on time complexity.

C.2 Space Complexity

The storage of the static vocabulary table takes up the majority of the total space storage. Therefore, the space complexities of BERT, BERM, and BERM-O are the same, i.e., O($n_w d$).

D Details of Experimental Setups

D.1 Baselines

The model BERM is compared with some state-of-the-art models. Like BERM, these models are used as the student model of the proposed knowledge dis-

tillation framework. We adopt the hyper-parameter settings recommended by the original papers for all the methods. According to the formulation process of embedding, these methods can be divided into the following three types:

- Three representation-focused relevance matching methods: DSSM (Huang et al., 2013), MVLSTM (Wan et al., 2016), and ARC-I (Hu et al., 2014b). They learn the low-dimension representations of both sentences w.r.t. a query and an item, and then predict their relationship by calculating the similarity (such as cosine similarity) of representations.

- Seven interaction-focused relevance matching methods: DRMM (Guo et al., 2016), MatchPyramid (Pang et al., 2016b), ARC-II (Hu et al., 2014a), K-NRM (Xiong et al., 2017), DRMM-TKS (Guo et al., 2016), Conv-KNRM (Xiong et al., 2017), and ESIM (Chen et al., 2017). They learn an interaction representation of both sentences based on the interaction calculation from word-level to sentence-level.

- Two integrated relevance matching methods: Duet (Mitra et al., 2017) and BERT2DNN (Jiang et al., 2020). They combine the features of the above two types of methods into themselves.

- Five graph neural network models: GAT (Veličković et al., 2018), GraphSAGE-Mean (Hamilton et al., 2017), GraphSAGE-LSTM (Hamilton et al., 2017), TextGNN (Zhu et al., 2021), and GEPS (Zhang et al., 2019). They aggregate the neighbor information from the query graph or item graph to update the embedding of the anchor node.

D.2 Implementation Details

Here we introduce the implementation details of the whole knowledge distillation framework as follows:

- **Teacher model**. For the teacher model, we adopt BERT-Base* with a 12-layer (k_1=12) Transformer encoder where the word embedding size d_1 is 768 and head number h_1 is 12. We pre-train BERT-Base on a human-labeled dataset with 380,000 query-item pairs. The fine-tuned BERT-Base is then used as an expert to refine the noisy click behavior data from S_{train}. The refinement rule is: if the prediction score y_i of BERT-Base is less than α (the default value of α is 0.3), then the raw edge is deleted; if the score is larger than β (the default value of β is 0.7), then a new edge is added.

- **Student model**. For the student model, we adopt the proposed BERM model. We implement BERM in TensorFlow 2.0 with the high-level Estimator API. For each input query phrase Q or item title I, we split it into several words and then truncate or pad its length to 10 or 65words (i.e., l_Q=10, l_I=65). Each word embedding is acquired by the lookup operation on a static vocabulary table whose total size n_w is 39,846. This table is generated by pre-training two billion search data with the tool of Word2Vec. The size d of pre-trained embeddings or trained embeddings is 128.

- **Training details**. We use Lazy-Adam as the optimizer and its learning rate is 0.001. To reduce the overfitting of the training data, we use L2 regularization on each layer of neural networks. For Data-E, we set the training epoch as 20. For Data-A and Data-S, we set the training epoch as 3.

E Additional Experiments

We conduct some additional experiments, including ablation studies (Section E.1) and sensitivity analysis (Section E.2). In these experiments, we adopt Data-E and Data-A consistently.

E.1 Ablation Study

E.1.1 Integration of Embeddings

There are three types of components in the complete BERM: the representation-based embeddings $\boldsymbol{E}^Q_{\text{seq}}$, $\boldsymbol{E}^I_{\text{seq}}$, interaction-based embedding $\boldsymbol{E}_{\text{int}}$, and metapath embeddings $\boldsymbol{E}_{Q\text{-}I\text{-}Q}$, $\boldsymbol{E}_{I\text{-}Q\text{-}I}$. To further examine the importance of each component in the final embedding of BERM, we remove one or two components from it (Equation 9) at a time and examine how the change affects its overall performance.

The corresponding results on Data-E and Data-A are reported in Table 5 and 6. We have the following empirical observation and analysis:

- In general, the both-component setting outperforms the single-component setting but is worse than the triple-component setting (i.e., BERM).

*https://github.com/google-research/bert

Table 5: Ablation study on Data-E. In each column, the best result is bolded.

Model	Data-E AUC	F1-score	FNR(↓)
$\boldsymbol{E}^{Q}_{\text{seq}}, \boldsymbol{E}^{I}_{\text{seq}}$	0.8537	0.8044	0.3560
$\boldsymbol{E}_{\text{int}}$	0.8595	0.8037	0.3464
$\boldsymbol{E}_{Q\text{-}I\text{-}Q}, \boldsymbol{E}_{I\text{-}Q\text{-}I}$	0.8430	0.8173	0.2995
$\boldsymbol{E}^{Q}_{\text{seq}}, \boldsymbol{E}^{I}_{\text{seq}}, \boldsymbol{E}_{\text{int}}$	0.8638	0.8086	0.3331
$\boldsymbol{E}_{\text{int}}, \boldsymbol{E}_{Q\text{-}I\text{-}Q}, \boldsymbol{E}_{I\text{-}Q\text{-}I}$	0.8761	0.8221	**0.2758**
$\boldsymbol{E}^{Q}_{\text{seq}}, \boldsymbol{E}^{I}_{\text{seq}}, \boldsymbol{E}_{Q\text{-}I\text{-}Q}, \boldsymbol{E}_{I\text{-}Q\text{-}I}$	0.8656	0.8190	0.2922
BERM	**0.8785**	**0.8256**	0.2966

Table 6: Ablation study on Data-A. In each column, the best result is bolded.

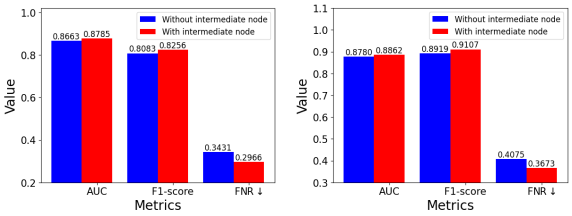
Model	Data-A AUC	F1-score	FNR(↓)
$\boldsymbol{E}^{Q}_{\text{seq}}, \boldsymbol{E}^{I}_{\text{seq}}$	0.8067	0.8660	0.3697
$\boldsymbol{E}_{\text{int}}$	0.8289	0.9084	0.4459
$\boldsymbol{E}_{Q\text{-}I\text{-}Q}, \boldsymbol{E}_{I\text{-}Q\text{-}I}$	0.8500	**0.9114**	0.4110
$\boldsymbol{E}^{Q}_{\text{seq}}, \boldsymbol{E}^{I}_{\text{seq}}, \boldsymbol{E}_{\text{int}}$	0.8776	0.9070	0.4743
$\boldsymbol{E}_{\text{int}}, \boldsymbol{E}_{Q\text{-}I\text{-}Q}, \boldsymbol{E}_{I\text{-}Q\text{-}I}$	0.8824	0.9099	0.3705
$\boldsymbol{E}^{Q}_{\text{seq}}, \boldsymbol{E}^{I}_{\text{seq}}, \boldsymbol{E}_{Q\text{-}I\text{-}Q}, \boldsymbol{E}_{I\text{-}Q\text{-}I}$	0.8750	0.9094	0.3753
BERM	**0.8862**	0.9107	**0.3673**

It demonstrates that different components in BERM have different positive effects on the overall performance and they cannot replace each other.

- The introduction of k-order relevance modeling can bring stable advancement to each 0-order relevance model. For example, the combination of "$\boldsymbol{E}^{Q}_{\text{seq}}, \boldsymbol{E}^{I}_{\text{seq}}, \boldsymbol{E}_{Q\text{-}I\text{-}Q}, \boldsymbol{E}_{I\text{-}Q\text{-}I}$" surpasses the combination of "$\boldsymbol{E}^{Q}_{\text{seq}}, \boldsymbol{E}^{I}_{\text{seq}}$" 6.83% according to the metric of AUC on Data-A. This demonstrates that applying metapath embedding to relevance matching can make effective use of the neighboring nodes' information in the user behavior graph.

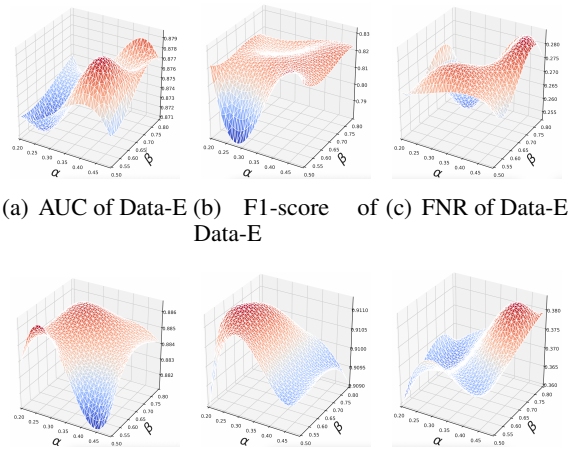
E.1.2 Effect of the Intermediate Node

The metapath defined in BERM includes the intermediate node. To further investigate the effect of the intermediate node, we compare the performances of BERM with the intermediate node (i.e., "Q-I-Q" and "I-Q-I") and BERM without the intermediate node (i.e., "Q-Q" and "I-I") on Data-E and Data-A in Figure 4. We observe that BERM with the intermediate node performs better than the other one. We infer that the intermediate node has strong semantic closeness to the anchor node and thus it is helpful for accurate semantic recognition.



(a) Data-E (b) Data-A

Figure 4: Effect of the intermediate node. The red (blue) bar represents BERM with (without) the intermediate node.



(a) AUC of Data-E (b) F1-score of Data-E (c) FNR of Data-E

(d) AUC of Data-A (e) F1-score of Data-A (f) FNR of Data-A

Figure 5: Effect of different values of α and β. (a), (b), and (c) are the results of Data-E; (d), (e), and (f) are the results of Data-A. The red (blue) color corresponds to the high (low) value.

E.2 Sensitivity Analysis

E.2.1 Thresholds α and β

In BERM, α decides how many edges of the noisy click behavior should be deleted and β decides how many hidden useful edges should be retrieved. To investigate the sensitivity of α and β, we conduct experiments with 16 different hyper-parameter settings where α ranges from 0.2 to 0.5 and β ranges from 0.5 to 0.8. We apply the three-order curve interpolation method to show the final results in Figure 5. In general, the results of BERM are robust to the change of hyper-parameter α and β on either Data-E or Data-A. For example, the maximum error of AUC is no more than 1%. So we conclude that user behaviors play a major role in the performance of BERM and the knowledge from BERT provides auxiliary effects for it.

Table 7: Effect of different neighbor selection strategies on Data-E.

Rate	Click+BERT's score			Purchase+BERT's score		
	AUC	F1-score	FNR(↓)	AUC	F1-score	FNR(↓)
$\lambda = 0.0$	**0.8785**	**0.8256**	0.2966	**0.8785**	**0.8256**	0.2966
$\lambda = 0.2$	0.8779	0.8237	**0.2834**	0.8777	0.8236	**0.2810**
$\lambda = 0.4$	0.8770	0.8200	0.3198	0.8756	0.8214	0.3068
$\lambda = 0.6$	0.8670	0.8085	0.4000	0.8696	0.8045	0.3694
$\lambda = 0.8$	0.8679	0.8101	0.3901	0.8660	0.8105	0.3262
$\lambda = 1.0$	0.8656	0.8091	0.3850	0.8671	0.8109	0.3727

Table 8: Effect of different neighbor selection strategies on Data-A.

Rate	Click+BERT's score			Purchase+BERT's score		
	AUC	F1-score	FNR(↓)	AUC	F1-score	FNR(↓)
$\lambda = 0.0$	**0.8862**	0.9107	**0.3673**	**0.8862**	0.9107	**0.3673**
$\lambda = 0.2$	0.8849	**0.9113**	0.3794	0.8830	**0.9117**	0.3831
$\lambda = 0.4$	0.8821	0.9112	0.4016	0.8818	0.9100	0.4131
$\lambda = 0.6$	0.8779	0.9068	0.4793	0.8783	0.9064	0.4844
$\lambda = 0.8$	0.8802	0.9076	0.4676	0.8790	0.9084	0.4683
$\lambda = 1.0$	0.8792	0.9077	0.4671	0.8787	0.9079	0.4716

E.2.2 Threshold k

Here we evaluate the effect of k on the performance of BERM by sampling neighboring nodes with different hops from the bipartite graph. The comparison results on Data-E and Data-S are shown in Figure 6. We can see that the BERM with k=2 achieves the best performance among them. When k is too large such as k=5, many distant neighbors are aggregated into the anchor nodes, then it leads to the performance degradation of BERM due to lots of noise gathering in these distant neighbors. Therefore, we conclude that 2-order relevance matching modeling is the optimal choice for our e-commerce scene.

E.2.3 Selection of Neighbor Structure

In BERM, the selection of neighbor structure directly affects which context information is transmitted to the anchor node. A good selection strategy can aggregate valuable neighboring node information to enrich the anchor node's representation. To investigate the effect of different neighbor structure selection strategies on BERM and seek a relatively optimal solution, we use different values of hyper-parameter λ to control the ratio between user behavior and BERT's score. Specifically, we calculate a new score $\text{Score}_{\text{new}}(Q, I) = \lambda \cdot \text{User}(Q, I) + (1-\lambda) \cdot \text{Score}(Q, I)$ where $\text{User}(Q, I)$ is the user behavior feature (e.g., for click behavior, $\text{User}(Q, I)$=1 if click behavior happens between query Q and item I). The addition and deletion of edges refer to $\text{Score}_{\text{new}}(Q, I)$, rather than $\text{Score}(Q, I)$. We report the results with different λ in Table 7 and 8. From them, we can conclude that:

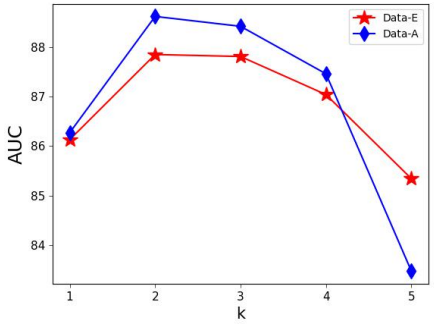


Figure 6: The effect of different k.

- Using BERT's score is better than using user behaviors for the selection of neighbors. Therefore, the value of AUC or F1-score gradually decreases with the increase of λ; the value of FNR increases with the increase of λ. The optimal λ is located in the interval $[0.0, 0.2]$.
- According to the metric of FNR, the purchase behavior is better than the click behavior on Data-E. The reason for it is that purchase behaviors reveal more accurate semantic relevance information than click behaviors. However, the purchase behavior is worse than the click behavior on Data-A. We guess that it is caused by the sparsity of purchase behaviors in the dataset of all categories.