RETE: Retrieval-Enhanced Temporal Event Forecasting on Unified Query Product Evolutionary Graph

Ruijie Wang[1*], Zheng Li[2†], Danqing Zhang[2], Qingyu Yin[2], Tong Zhao[2], Bing Yin[2] and Tarek Abdelzaher[1]
[1]University of Illinois at Urbana Champaign, IL, USA [2]Amazon.com Inc, CA, USA
{ruijiew2,zaher}@illinois.edu,{amzzhe,danqinz,qingyy,zhaoton,alexbyin}@amazon.com

ABSTRACT

With the increasing demands on e-commerce platforms, numerous user action history is emerging. Those enriched action records are vital to understand users' interests and intents. Recently, prior works for user behavior prediction mainly focus on the interactions with product-side information. However, the interactions with search queries, which usually act as a bridge between users and products, are still under investigated. In this paper, we explore a new problem named temporal event forecasting, a generalized user behavior prediction task in a unified query product evolutionary graph, to embrace both query and product recommendation in a temporal manner. To fulfill this setting, there involves two challenges: (1) the action data for most users is scarce; (2) user preferences are dynamically evolving and shifting over time. To tackle those issues, we propose a novel *Retrieval-Enhanced Temporal Event* (RETE) forecasting framework. Unlike existing methods that enhance user representations via roughly absorbing information from connected entities in the whole graph, RETE efficiently and dynamically retrieves relevant entities centrally on each user as high-quality subgraphs, preventing the noise propagation from the densely evolutionary graph structures that incorporate abundant search queries. And meanwhile, RETE autoregressively accumulates retrieval-enhanced user representations from each time step, to capture evolutionary patterns for joint query and product prediction. Empirically, extensive experiments on both the public benchmark and four real-world industrial datasets demonstrate the effectiveness of the proposed RETE method.

CCS CONCEPTS

• **Information systems** → **Electronic commerce**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

Temporal Event Forecasting, Dynamic Graph Learning, E-commerce

*Part of work was done during internship at Amazon; †Corresponding author.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3511974

ACM Reference Format:
Ruijie Wang, Zheng Li, Danqing Zhang, Qingyu Yin, Tong Zhao, Bing Yin and Tarek Abdelzaher. 2022. RETE: Retrieval-Enhanced Temporal Event Forecasting on Unified Query Product Evolutionary Graph. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3485447.3511974

1 INTRODUCTION

On shopping websites or platforms, users type search queries and then perform various actions on the returned products. Such behaviors produce numerous interactions to both search queries and products, such as *"type a search query"*, *"click a product page"*, *"add a product to the shopping cart"* or *"purchase a product"*. Jointly modeling user-query and user-product interactions are essential to identifying users' preferences and intents for reasonable and interpretable behavior prediction.

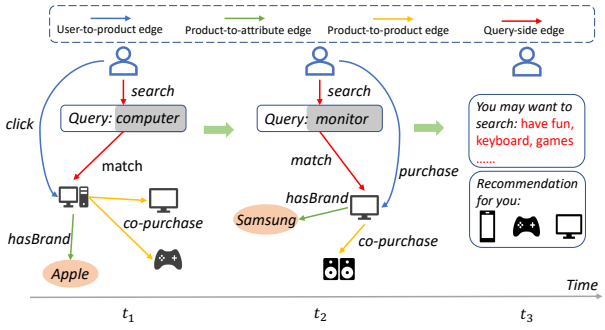


Figure 1: Illustration of temporal event forecasting on the E-commerce search domain. A set of interactions associated with temporal information between users, queries, products, and attributes constitute to user events at each time. Existing methods mainly focus on product-side interactions (Blue, Yellow, Green lines) while ignoring the underlying influence from search queries (Red line) explored in our work.

Unfortunately, as shown in Figure 1, to predict user instant behaviors, existing works merely focus on product-side interactions (e.g, user-to-product [9, 14, 23, 32], product-to-product [13, 17, 37], or product-to-attribute [4, 40, 44, 52]) but neglect equally important search queries. This usually causes incomplete and noisy user profile characterization, especially in the cold-start regime that lacks sufficient contexts. Concretely, as one crucial information carrier from human natural languages, search queries act like a bridge to link users and products. On one hand, users utilize search queries to express shopping intents, which are usually the prerequisite for

subsequent user-product interactions. So queries can be directly closer to users' intents and suffice to reliably capture underlying interests. On the other hand, jointly modeling query and product information can bring in mutual benefits. For example, for users who accidentally interact with unrelated products in the imperfect matching sets returned by search engines, query information can be utilized to denoise those abnormal behaviors, and vice verses.

To embrace the collective power from both products and queries, we generalize user behavior prediction as a new problem setting named *Temporal Event Forecasting*, where a diverse range of interactions associated with temporal information between users, queries, products, and attributes constitute to a large-scale unified query product evolutionary graph. Our goal aims to dynamically capture user interests' dependency and evolution on this temporal knowledge graph by predicting user events at future times, where the event prediction can be disentangled as simultaneously predicting potential search queries and interacted products.

The challenges in fulfillment of the proposed setting are twofold: (1) *scarce action data*: the limited action records of most users, due to the long-tail distribution [12], make it hard to obtain robust user representations. To tackle the issue, prior works stack multiple aggregation layers [38, 40, 44], e.g., graph neural networks (GNNs) [21], to accumulate information from other entities such as product side information [4, 44] or similar users [40], to enhance user representations. However, as number of propagation layers grows, the neighborhood size increases exponentially, especially in a more connected graph like ours in light of abundant search queries. This can usually bring in extensive noises from unrelated and repeated entities, making user representations indiscriminative (a.k.a over-smoothing) [26, 50, 54]; (2) *interest evolution*: user preferences and intents are dynamically evolving and shifting over time. As new user activities are continuously emerging and collected, such new interaction events, reflecting users' most recent intents, may have large distribution gaps with previous user action behaviors. Failing to model such time-varing patterns usually results in significant performance degradation as time goes by [43, 53], especially for modeling the joint objective of query and product.

Motivated by those challenges, we propose a retrieval-enhanced temporal event forecasting framework named *RETE*, to learn both discriminative and temporally-aware user representations for joint query and product recommendation. To enrich each user profile, RETE exploits subgraph samplers to dynamically filter out unrelated noises and retrieve higher-order entities centrally on each user from dense and temporal graph structures. Those retrieved entities are organized as subgraphs and integrated to enhance each user representation via a structural attention module. As such, the information propagation is preserved within the local related structures from high-quality subgraphs instead of the noisy global graph. To better capture user intent evolution over time, a sequence of retrieval-enhanced user representations from each time step are accumulated in an autoregressive manner via a temporal attention module, which can automatically learn the importance of user interaction events distributed in the axis of time. This adapts retrieval-enhanced user representation to be time-aware for capturing evolutionary patterns.

To validate the effectiveness of RETE, we conduct extensive experiments on both the public Yelp Challenge 2019 dataset and four real-world large-scale industrial datasets. Our experimental results demonstrate the superiority of RETE over state-of-the-art baselines by a large margin, including factorization machine (FM-based) [14, 32], session-based [17, 37], knowledge graph (KG-based) [4, 44] recommendation models as well as dynamic graph learning methods [23]. We further conduct comprehensive ablation and hyperparameter studies to validate the effectiveness of each design choices. Finally, we demonstrate the insights towards prediction interpretability by visualizing temporal weights across time steps.

Table 1: Comparison with existing settings. Action history refers to various past users' behaviors towards products such as "Add", "Click" and "Purchase", etc. Search history denotes historical user search queries. Meta data refers to product side information like attributes.

Properties		Action history	Search history	Meta data	Temporal Info.	Multi objective
Recommendation	FM-based [14, 32]	✓	✗	✓	✗	Product
	Sequential / Session-based [11, 17, 31, 36, 37]	✓	✗	✗	✓	Product
	KG-based [4, 40, 44, 52]	✓	✗	✓	✗	Product
Query suggestion [1, 2]		✗	✓	✗	✗	Query
Dynamic Graph Learning [9, 23]		✓	✗	✗	✓	Product
Temporal Event Forecasting (**Ours**)		✓	✓	✓	✓	Product & Query

Overall, our contributions can be summarized in follows:

- **Problem formulation**: To the best of our knowledge, our work is the first attempt to propose a unified setting named temporal event forecasting, considering both query and product oriented prediction in a temporal manner.
- **Novel framework**: We propose a novel RETE framework to enhance the temporal event prediction for low-data users and meanwhile alleviate the over-smoothing issue by dynamically retrieving user-centered entities from the highly-connected query product evolutionary graph.
- **Extensive evaluations**: Empirically, extensive experiments on both the public dataset and large-scale industrial datasets demonstrate the effectiveness of the proposed RETE method.

2 RELATED WORK

We discuss and compare existing works from three related areas, as summarized in Table 1.

Product recommendation. Recommendation systems (RS) have achieved huge success on E-commerce platforms. Numerous efforts are devoted to improve RS from different perspectives: factorization machine (FM-based) models [14, 24, 32] efficiently consider abundant input features, sequential/session-based recommendation models [11, 17, 30, 31, 36, 37] capture user-side dynamics from long/short time periods respectively, and knowledge graph (KG-based) [4, 40, 44] models consider higher-order connections in multi-relational graph and produce explainable prediction. However, existing works merely focus on product-side interactions (e.g, user-to-product [9, 14, 23, 32], product-to-product [13, 17, 37], or product-to-attribute [4, 40, 44, 52]) but neglect equally important search queries. We study a more generic temporal event forecasting task to jointly optimize product and query prediction on evolutionary knowledge. A few interactive recommendation works on knowledge graph optimize sequential policy for recommending

products within short sessions [55]. In contrast, we focus our attention on studying how to better integrate time-aware information from evolutionary knowledge graph and exclude unrelated noises in order to improve long-term performance for both product prediction and query prediction.

Query suggestion. Query suggestion task aims to assistant users to better express their intents on various search engine systems. General query suggestion task includes three different objectives: query rewriting (QR) [6, 16], query auto-completion (QAC) [27, 28] and query prediction [1, 2]. While QR and QAC focus on reformulating the queries that help the users to find better search results of current search intents, we specifically study query prediction that "recommend" queries that potentially match user intents. Existing works on recommending/predicting queries utilize search log data [1], query dependency graph [8] or interaction history with users [2]. Instead, to the best of our knowledge, we are the first to propose and study joint product and query prediction task on E-commerce. Inspired by the recent success of combining both queries and documents for jointly information retrieval in search engine systems [3], we mainly focus on exploring mutual benefits from both queries and products. One major difference is that we only consider structured graph data, instead of content, as query contents are much more private and sensitive on E-commerce.

Dynamic graph learning. Graph learning methods [10, 19, 21, 29, 47, 49] has been widely explored for recommendation [15, 44], user modeling [35, 39, 42, 46, 48], etc. Recently, dynamic algorithms are proposed to better capture the temporal patterns of evolutionary graph [10, 20, 23, 33, 41]. As the constructed query product evolutionary graphs are more diverse and complicated, it brings new challenges to deal with over-smoothing issues and to capture temporal behavior patterns at the same time. Hence we aim to design a dynamic model with subgraph samplers to better learn informative and time-aware user representations. Although existing works have explored combinational power of subgraph samplers with graph neural networks on static and homogeneous/bipartite graph [26, 51], we further facilitate dynamic graph learning with subgraph samplers on the evolutionary knowledge graph.

3 PRELIMINARIES AND ANALYSIS

3.1 Concepts and Notations

We study query-centric events and product-centric events associated with temporal information, i.e., a user u types a search query q or performs actions on a product p at the time t. A set of event records constitutes to the dynamic interaction graph $\mathcal{G}_A^t = \{(e, r, e', t) | e \in \mathcal{U}, e' \in \mathcal{Q} \bigcup \mathcal{P}, r \in \mathcal{R}_A\}$, where $\mathcal{U}, \mathcal{Q}$ and $\mathcal{P}$ denote the user set, query set and product set, respectively. Relation set $\mathcal{R}_A$ represents the interaction types among them, such as *"type the query"*, *"click the product"*, *"adding product to carts"* and so on.

At the same time, rich meta-data of products forms a heterogeneous product graph $\mathcal{G}_P$, describing important attributes of each product $p \in \mathcal{P}$. Specifically, $\mathcal{G}_P = \{(e, r, e') | e \in \mathcal{P}, e' \in \mathcal{I} \bigcup \mathcal{Q}, r \in \mathcal{R}_P\}$, where $\mathcal{I}$ denotes attribute set for products, including but not limited to brand, product type and category. $\mathcal{R}_P$ denotes the relation set among them. $\mathcal{G}_P$ also describes match relations between products and queries.

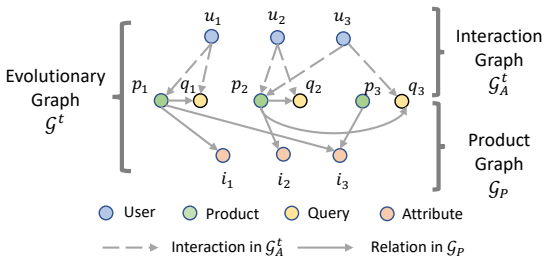


Figure 2: Query Product Evolutionary Graph at the t-th time.

To capture the temporal patterns of user behaviors, we split time period into discrete time steps $t = 1, 2, \cdots, T$. Within each time step, for convenience of discussion, we unify the dynamic interaction graph $\mathcal{G}_A^t$ and the product graph $\mathcal{G}_P$ as a snapshot of *evolutionary knowledge graph*: $\mathcal{G}^t = \mathcal{G}_A^t \bigcup \mathcal{G}_P$, as shown in Figure 2.

3.2 Definition

*Definition 3.1 (**Temporal event forecasting**).* Given a collected evolutionary knowledge graph $\mathbb{G} = \{\mathcal{G}^1, \mathcal{G}^2, \cdots, \mathcal{G}^T\}$, for each user $u \in \mathcal{U}$, temporal event forecasting aims to predict potentially interacted query $q \in \mathcal{Q}$ and product $p \in \mathcal{P}$ after T.

3.3 Analysis

On the evolutionary knowledge graph, user intents can be captured by integrating abundant information from connected entities [52]. Existing KG-based frameworks [4, 40, 44] map entities into a low-dimensional space, such that the relevance between user, query or product can be modeled via corresponding representations, i.e., $\boldsymbol{h}_u, \boldsymbol{h}_p, \boldsymbol{h}_q \in \mathbb{R}^d$. They propose various propagation mechanisms [21, 38] on whole KG to integrate abundant information for each user, which can be generally described as follows:

$$\boldsymbol{h}_u^{(l)} = \sum_{e' \in \mathcal{N}_u} \pi(u, r, e') \boldsymbol{h}_{e'}^{(l-1)}, \tag{1}$$

where $\mathcal{N}_u$ denotes neighbor set of user u, l denotes the number of propagation layers, triple (u, r, e') describes interaction between u and e', and $\pi(u, r, e')$ denotes aggregation weights.

Directly generalizing this family of propagation mechanisms to the event forecasting task, especially on evolutionary knowledge graph, faces two issues: (i) As l grows to integrate higher-order information, the neighborhood size increases exponentially. A large ratio of unrelated entities (noises) are integrated, making user representations less distinguishable from each other, or even leading to over-smoothing [26, 50, 54]; (ii) evolutionary knowledge graph shows different distribution over time as users have evolving intents and behavior patterns. Ignoring such temporal factors not only fails to capture the most recent data characteristics but also worsens the first issue due to integrating a large ratio of out-of-fashion records for learning the representations.

4 METHODOLOGY

In this section, we present the proposed RETE framework. We firstly start with the overview of RETE and then detail three major components. Finally, we describe the model optimization.

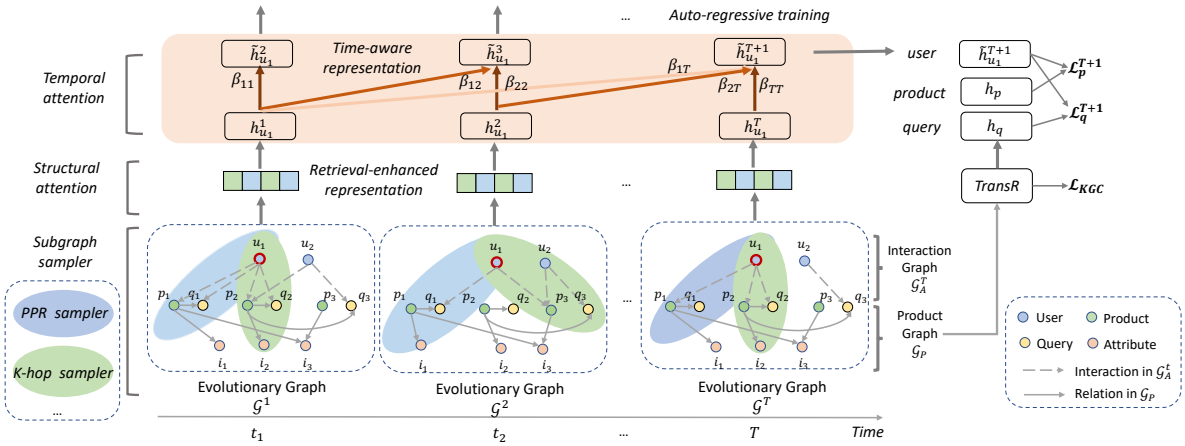


Figure 3: The framework of the proposed RETE model. We optimize the RETE model using the auto-regressive ranking loss and the knowledge graph completion loss (TransR) alternatively.

4.1 Overview

To address two issues mentioned in Section 3.3 simultaneously, the proposed RETE strives to satisfy the following requirements:

- *Requirement 1*: It should learn informative and discriminative user representations by considering higher-order information and filtering out large ratio of noise from the graph.
- *Requirement 2*: It should capture user intent evolution from data at different time steps, so as to produce up-to-date forecasting.

The framework of RETE is shown in Figure 3. Based on the collected evolutionary knowledge graph, the key idea of RETE is to learn informative user representations at each time step from the sampled subgraphs instead of the whole graph, and combine them together via learnable temporal weights to capture user intent evolution. RETE first utilizes subgraph samplers to retrieve related entities centrally around each user u. And then it integrates rich information from the subgraphs via a structural attention module. A sequence of learned representations from all time steps are integrated via a temporal attention module. It can automatically learn the combination weights in the temporal domain so that more related (e.g., more recent) time steps are assigned larger weights, and unrelated (e.g., far away) time steps are assigned smaller weights (but not necessary to 0, as they may also reflect long-term intents).

To fulfill *Requirement 1*, the sampled subgraphs constrain the attentive information propagation within local highly-related structure instead of the whole graph, so as to consider higher-order information and filter out noise. The design of the temporal module enables us to auto-regressively learn users' up-to-date representations to meet *Requirement 2*. We refer readers to Appendix A.1 for theoretical analysis of how RETE satisfies two requirements. Next, we will introduce the subgraph samplers, the structural attention module, and the temporal attention module respectively.

4.2 Ensemble subgraph sampler

The proposed sampler aims to retrieve diverse and related entities via higher-order connections and excludes unrelated noises as order increases. To design strong indicators for such desirable subgraphs, we adopt structure-dependent Personalized PageRank (PPR) value [18] which has been recently shown effective to retrieve related nodes from homogeneous graph [7, 22, 51]. And we further ensemble it with a simple randomized k-hop sampler to retrieve entities more comprehensively. Advantages of this design, instead of utilizing trainable policy, are that it does not require reliable input features (unlike [26], it is fully feature-independent) and it achieves much higher sample efficiency (almost real-time with careful implementation, as shown in Appendix A.3). We summarize the designed sampler as follows:

- **PPR sampler**. We use the feature-independent Personalized PageRank (PPR) value. Given a target user u, our PPR sampler first computes the approximate PPR value for all other entities, then selects up to b neighborhood above threshold θ and preserves relations among selected entity set.
- **k-hop sampler**. Starting from a target user u, the k-hop sampler traverses up to k-hop connections and randomly selected up to b neighbors.
- **Ensemble sampler**. To capture a full picture of user intents, we ensemble multiple samplers under different types or with different parameters to parallelly sample several subgraphs.

4.3 Structural attention module

Without loss of generality, let s denote the number of subgraphs from the ensemble sampler. At time step t, given the sampled subgraphs $\{\mathcal{G}^{1t}_{[u]}, \cdots, \mathcal{G}^{st}_{[u]}\}$ for each user u, the structural attention module aims to integrate all useful information to learn user representations. As shown in Figure 4, it first extracts information from each subgraph via multi-layer graph attentions [38]. Then to combine information from different perspectives, it fuses outputs from different subgraphs together to capture the global picture of user intent.

Specifically, we first utilize L graph attention layers to propagate and integrate information within each subgraph $\mathcal{G}^{st}_{[u]}$. Each layer can be summarized in Eq. 2:

$$\boldsymbol{h}_u^{(l)} = \sigma\left(\sum_{v \in \mathcal{N}_u} \alpha_{uv}^{(l)} \boldsymbol{W}_V^{(l)} \boldsymbol{h}_v^{(l-1)}\right), (1 \le l \le L) \tag{2}$$

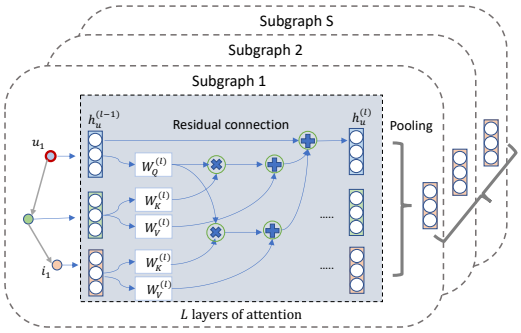


Figure 4: Structural attention module on subgraphs.

where $\boldsymbol{h}_u^{(l)}$ denotes user representations from layer l, $\mathcal{N}_u$ denotes neighbor set around user u on subgraph $\mathcal{G}_{[u]}^{st}$, and $\alpha_{uv}^{(l)}$ is aggregation attention weight shown in following equation:

$$\alpha_{uv}^{(l)} = \frac{\exp\left(\sigma\left(\boldsymbol{a}^T\left[\boldsymbol{W}_Q^{(l)}\boldsymbol{h}_u^{(l)} \| \boldsymbol{W}_K^{(l)}\boldsymbol{h}_v^{(l)}\right]\right)\right)}{\sum_{v' \in \mathcal{N}_u}\exp\left(\sigma\left(\boldsymbol{a}^T\left[\boldsymbol{W}_Q^{(l)}\boldsymbol{h}_u^{(l)} \| \boldsymbol{W}_K^{(l)}\boldsymbol{h}_{v'}^{(l)}\right]\right)\right)}, \tag{3}$$

where $\boldsymbol{W}_V^{(l)}, \boldsymbol{W}_K^{(l)}, \boldsymbol{W}_Q^{(l)}$ are shared weighted transformation applied to each entity in the subgraph, $\boldsymbol{a}$ is a weight vector parameterizing the attention function implemented as feed-forward layer, $\|$ is the concatenation operation and $\sigma(\cdot)$ is non-linear activation function.

Since information propagation is constrained within the sampled subgraphs, more attention layers can be stacked to better learn latent representations without introducing other unrelated entities and noises. We then use residual connection across L layers and graph-level pooling to better integrate information to capture user's intent at time step t:

$$\boldsymbol{h}_u^{st} = \frac{1}{|\mathcal{E}_{[u]}^{st}| \times L}\sum_{e \in \mathcal{E}_{[u]}^{st}}\sum_{l}\boldsymbol{h}_u^{(l)}, \tag{4}$$

where $\mathcal{E}_{[u]}^{st}$ denotes entity set in s-th subgraph $\mathcal{G}_{[u]}^{st}$. By doing so, we are able to integrate all useful information from the subgraphs to better represent users. To further capture a global view of user intent, ensemble samplers sample several subgraphs. From the learned representations of those subgraphs, we utilize one-layer MLP to integrate global information:

$$\boldsymbol{h}_u^t = \sigma\left(\boldsymbol{W}\left[\boldsymbol{h}_u^{1t} \| \boldsymbol{h}_u^{2t} \| \cdots \| \boldsymbol{h}_u^{st}\right]\right). \tag{5}$$

4.4 Temporal attention module

Given the learned user representations $\boldsymbol{H} = \{\boldsymbol{h}_u^1, \boldsymbol{h}_u^2, \cdots, \boldsymbol{h}_u^T\}$ from all time steps, we infer users' intents in a near future, i.e., $\widetilde{\boldsymbol{h}}_u^{T+1}$, for temporal event forecasting. We propose a temporal attention module to automatically capture both long-term and short-term intents by assigning different weights among h_u^t. The updating function for most recent user representation $\boldsymbol{h}_u^{T+1}$ can be expressed as follows:

$$\widetilde{\boldsymbol{h}}_u^{T+1} = \sum_{t=1}^{T}\beta_{tT}\boldsymbol{h}_u^t\boldsymbol{W}^V, \tag{6}$$

$$\beta_{ij} = \text{softmax}\left(\frac{\boldsymbol{H}\boldsymbol{W}^Q(\boldsymbol{H}\boldsymbol{W}^K)^T}{\sqrt{d}} + \boldsymbol{M}\right)_{ij}, \tag{7}$$

where $\boldsymbol{W}^Q$, $\boldsymbol{W}^L$, $\boldsymbol{W}^V$ are trainable temporal parameters, β_{ij} is learned temporal weight, d denotes dimension of user representations, and $\boldsymbol{M}$ is added to ensure auto-regressive setting, i.e., preventing future information affecting current state. We define $\boldsymbol{M}_{ij} = 0$ if $i \le j$, otherwise $-\infty$.

By applying Eq. 6, we are able to not only emphasize those information related to users' short-term intents represented in $\boldsymbol{h}_u^T$, but also capture long-term intents as we integrate information from all time steps. The proposed temporal module provides better interpretability, where the temporal attention weights reflect user intent evolution and shifting. Notably, it can be auto-regressively applied to newly collected data from $T+2, T+3, \cdots$, as it automatically computes temporal combination weights in the future time steps without model modification or retraining.

4.5 Optimization

The proposed framework is expected to capture the evolution of user preference from the evolutionary knowledge graph. To better represent product and query information from static product graph into the same low-dimensional space, we utilize a knowledge graph embedding module TransR [25] to optimize knowledge graph completion loss $\mathcal{L}_{KGC}$, which is detailed in Appendix A.2. After learning user/product/query representations at time step t, i.e., $\widetilde{\boldsymbol{h}}_u^t$, $\boldsymbol{h}_p, \boldsymbol{h}_q$, we use inner product $\gamma_{up}^t =< \widetilde{\boldsymbol{h}}_u^t, \boldsymbol{h}_p >$ and $\gamma_{uq}^t =< \widetilde{\boldsymbol{h}}_u^t, \boldsymbol{h}_q >$ to model relevance.

We use negative sampling to accelerate and stabilize the training process. At time t, for each user-product pair (u, p^+), we randomly sample several negative samples (u, p^-), where we expect γ_{up-}^t is smaller than γ_{up+}^t by a margin. Thus, we adopt the weighted approximate-rank pairwise (WARP) loss [45] for product prediction as follows:

$$\mathcal{L}_p = \sum_{t=1}^{T}\mathbb{E}_{(u,p^+)\in\mathcal{G}^t}\sum_{p^-}\frac{L(rank(p^+)) \cdot |\lambda_m - \gamma_{up+}^t + \gamma_{up-}^t|_+}{rank(p^+)}, \tag{8}$$

where λ_m denotes margin value, $|\cdot|_+$ means $\max(0, \cdot)$. For each observed interaction (u, p^+), we expect the relevance score γ_{up+}^t to be larger than that of any negative samples by λ_m, i.e., $|\lambda_m - \gamma_{up+}^t + \gamma_{up-}^t|_+ = 0$. Otherwise, we penalize each pair of (p^+, p^-) because of the incorrect ranking. $L(K) = \sum_{k=1}^{K} 1/k$, $rank(p^+)$ denotes relative ranking of positive sample p^+ among negative samples p^-, and $L(rank(p^+))$ is the penalty weight. Similarly we can define WARP loss for query prediction as $\mathcal{L}_q$. Thus, the overall objective is:

$$\mathcal{L} = \mathcal{L}_p + \mathcal{L}_q + \mathcal{L}_{KGC} + \|\Theta\|_2, \tag{9}$$

where Θ denotes model parameters, $\mathcal{L}_{KGC}$ denotes TransR loss to represent static facts in the product graph. We optimize $\mathcal{L}_{\text{KGC}}$ and $\mathcal{L}_p + \mathcal{L}_q + \|\Theta\|_2$ alternatively, where mini-batch Adam is adopted. We summarize the optimization process of RETE in Algorithm 1.

Algorithm 1: The optimization process for RETE.

Input: Evolutionary knowledge graph $\{\mathcal{G}^1, \cdots, \mathcal{G}^T\}$ and product graph $\mathcal{G}_P$.
Output: User intents $\widetilde{\boldsymbol{h}}^{T+1}$ and model parameters Θ.

```
Ensemble samplers sample subgraphs for users;
while model not converged do
    Optimize on product graph:
    Minimize L_KGC and update entity representation in G_P;
    Optimize on evolutionary graph:
    for Each time step t < T during training do
        (Auto-regressive training:)
        Learn user intent {h^1, ..., h^{t-1}} according to Eq. 5;
        Update new intents h̃^t according to Eq. 6;
        Calculate ranking loss L_p^t and L_q^t at time step t;
    end
    Optimizing ranking loss L_p and L_q in Eq. 8;
end
Update user intents h̃^{T+1} according to Eq. 6;
```

5 EXPERIMENT

We evaluate RETE on one public and four real-world E-commerce datasets [1], and we aim to answer the following research questions:

- **RQ1**: How does RETE perform compared with state-of-the-art models on the datasets in both academia and industry?
- **RQ2**: How do different components affect RETE performance?
- **RQ3**: Can RETE better integrate information from neighbors?
- **RQ4**: Can RETE capture the evolution of users' preferences?

5.1 Experimental setup

5.1.1 Datasets. We collected one public Yelp dataset and four industrial E-commerce datasets for experiments:

- **Yelp**. The dataset is adopted in Yelp Challenge 2019 [2], which contains the interaction records between users and businesses like restaurants and bars. For ease of evaluation, we extract data since April 2014, spanning a period of more than ~ 7 years. We generate pseudo queries by extracting representitive keyphrases from user reviews. And we remove users, products, queries with less interactions than 20. To construct product graph we use attributes like category, location, etc.
- **E-commerce**. We gain access to the search log data spanning a period of 140 days and product attribute data. We first collect data under four specific categories: *Electronics*, *Book*, *Music* and *Beauty*. For each categories, we retain users, products, queries with at least 10 interactions. Then to construct product graph, we preserve product attributes, including brand, product type, etc.

We purposefully choose the two platform because of various length of time range. To evaluate our framework, we divide time span into 28 time steps according to interaction timestamps. We split them into background/training/val/test (10/10/2/6) to train initial entity embeddings (model input), train, validate and test RETE respectively. We also try different time segmentation strategies, where our method consistently outperforms others. We leave a systematic study for optimal segmentation as our future work. Table 5 in Appendix A.4 summarizes the statistics of the experimental datasets.

5.1.2 Metrics. We evaluate temporal event forecasting task in a retrieval setting, i.e., we compare the predicted top-K ranking list of products/queries with the groundtruth in the testing time steps. We adopt two widely-used evaluation protocols: *Recall@K* and *NDCG@K*. By default, we set $K = 20$.

5.1.3 Baselines. We compare baselines from following areas:

- FM-based recommendation, which considers the second-order feature interactions. We compare **FM** [32] and **NFM** [14].
- Sequential recommendation, which considers user evolving intents overtime. We compare **GRU4Rec** [17] and **BERT4Rec** [37].
- KG-based recommendation, which models heterogeneous entities and high-order connections for recommendation. We compare **ECFKG** [4] and **KGAT** [44].
- dynamic graph learning: which models evolutionary interaction graph. We compare **JODIE** [23].

Details can be found in Appendix A.4, including data collection, data statistics, baseline/model setup, hyper-parameter tuning, etc.

5.2 Model Performance (RQ1)

5.2.1 Overall performance. We first compare overall performance of product prediction and query prediction with selected baselines, as shown in Table 2a and Table 2b. In most cases, FM-based (FM, NFM) and sequential recommendation (BERT4Rec, GRU4Rec) methods produce poor results, as they do not explicitly consider higher-order interactions. RETE beats KG-based methods (KGAT, ECFKG) and dynamic graph learning method (JODIE) on all metrics, as we propose a better way to integrate multi-relational data in a temporal manner. Notably, on E-commerce platform, query prediction has worse performance than predicting products, while Yelp platform exhibits different pattern. We hypothesis that it is because the real queries from users on E-commerce platform are more diverse than pseudo queries extracted from Yelp review data. Also, it is harder to produce accurate prediction on Yelp platform, as Yelp data are collected from much longer period, where user intent shifting and evolution across time step are much harder to capture.

5.2.2 Detailed performance. Further, to investigate how does RETE perform over time, we compare the detailed performances in each testing time step (6 time step), as shown in Figure 5. The performances in different time steps vary largely, indicating user intents are evolving and shifting. RETE can beat others in almost all time steps, and more significant improvements come from the last several time steps, which shows our proposed temporal module can capture the evolution of user preference and thus achieve better long-term performance.

5.2.3 Auto-regressive evaluation. As users keep interacting with E-commerce platforms, new interaction events are collected continuously. In real scenario, it is required that the deployed models

[1] Our code is open-source and available at https://github.com/amzn/RETE_WWW2020.
[2] https://www.yelp.com/dataset/

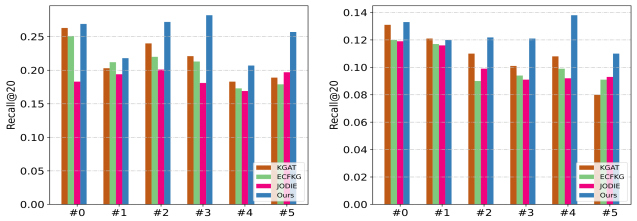
Table 2: Overall performance for product and query prediction. Average results on 5 independent runs are reported. $*$ indicates the statistically significant improvements over the best baseline, with p-value smaller than 0.001.

Dataset	Public		Industrial E-commerce							
	Yelp		Electronics		Music		Book		Beauty	
$K = 20$	NDCG@K	Recall@K	NDCG@K	Recall@K	NDCG@K	Recall@K	NDCG@K	Recall@K	NDCG@K	Recall@K
FM-based Recommendation										
FM	0.0221	0.0277	0.0512	0.0713	0.0641	0.0981	0.0682	0.0964	0.1155	0.1459
NFM	0.0214	0.0281	0.0715	0.1164	0.0761	0.1005	0.0793	0.1064	0.1246	0.1591
Sequential Recommendation										
BERT4Rec	0.0422	0.0501	0.0619	0.0832	0.0537	0.0618	0.0447	0.0651	0.0827	0.1015
GRU4Rec	0.0419	0.0511	0.0742	0.0859	0.0621	0.0711	0.0412	0.0658	0.0842	0.1003
Dynamic Graph Learning										
JODIE	0.0459	0.0527	0.1399	0.1515	0.1123	0.1405	0.1401	0.1881	0.1458	0.1807
KG-based recommendation										
KGAT	0.0342	0.0403	0.1503	0.1914	0.1156	0.1301	0.1254	0.1479	0.1503	0.1893
ECFKG	0.0388	0.0495	0.1413	0.1859	0.1036	0.1246	0.1327	0.1674	0.1401	0.1799
RETE (Ours)	**0.0499***	**0.0589***	**0.1703***	**0.2120***	**0.1304***	**0.1521***	**0.1455***	**0.1976***	**0.1621***	**0.1985***
Gain	*8.71%*	*11.76%*	*13.31%*	*10.76%*	*12.80%*	*8.27%*	*3.85%*	*5.05%*	*7.85%*	*4.86%*

(a) Product prediction performance.

Dataset	Public		Industrial E-commerce							
	Yelp		Electronics		Music		Book		Beauty	
$K = 20$	NDCG@K	Recall@K	NDCG@K	Recall@K	NDCG@K	Recall@K	NDCG@K	Recall@K	NDCG@K	Recall@K
FM-based Recommendation										
FM	0.0257	0.0319	0.0481	0.0765	0.0324	0.0681	0.0862	0.1015	0.0614	0.0854
NFM	0.0244	0.0331	0.0533	0.0709	0.0583	0.1188	0.0851	0.1103	0.0673	0.0903
Sequential Recommendation										
BERT4Rec	0.0407	0.0498	0.0602	0.0877	0.0207	0.0457	0.0413	0.0882	0.0417	0.0566
GRU4Rec	0.0381	0.0477	0.0590	0.0731	0.0436	0.0599	0.0401	0.0907	0.0513	0.0602
Dynamic Graph Learning										
JODIE	0.0461	0.0617	0.0779	0.0957	0.0988	0.1364	0.1301	0.1475	0.1327	0.1495
KG-based recommendation										
KGAT	0.0431	0.0527	0.0913	0.1153	0.0823	0.1324	0.1293	0.1497	0.1299	0.1502
ECFKG	0.0397	0.0481	0.0899	0.1099	0.0897	0.1259	0.1283	0.1503	0.1214	0.1518
RETE (Ours)	**0.0507***	**0.0653***	**0.1015***	**0.1393***	**0.1033***	**0.1408***	**0.1391***	**0.1557***	**0.1487***	**0.1643***
Gain	*9.98%*	*5.83%*	*11.17%*	*20.82%*	*4.55%*	*3.22%*	*6.92%*	*4.01%*	*12.06%*	*9.31%*

(b) Query prediction performance.



(a) Product. **(b) Query.**

Figure 5: Recall@20 of each test period on *Electronics*, significant improvements come from the last several time steps.

can take newly collected data to update user representations without time-consuming retraining or fine-tuning. We refer it to *auto-regressive* evaluation and verify the robustness of RETE under it. As it is hard to update static models on new data without retraining, we mainly focus on comparing with dynamic models (BERT4Rec, GRU4Rec and JODIE). Given new testing data, we continuously fed them into the temporal module and evaluate the performance in the next time step. Table 3 reports the average performance on testing time steps. All compared models achieve improved results after considering newly collected data, as which contain more up-to-date clues to capture users' intents. RETE can achieve the best performance, showing better generalization ability and robustness for continual learning.

Table 3: Performance under the auto-regressive evaluation. Average results on six testing time steps are reported.

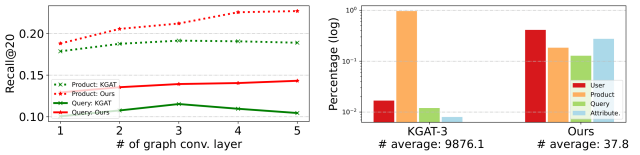
Task	Product prediction			
Dataset	Electronics		Music	
k = 20	NDCG@K	Recall@K	NDCG@K	Recall@K
BERT4Rec	0.0830	0.1232	0.0566	0.0701
GRU4Rec	0.1099	0.1201	0.0519	0.0803
JODIE	0.1801	0.1962	0.1371	0.1507
Ours	**0.1961**	**0.2414**	**0.1561**	**0.1733**
Task	Query prediction			
Dataset	Electronics		Music	
k = 20	NDCG@K	Recall@K	NDCG@K	Recall@K
BERT4Rec	0.0861	0.1019	0.0455	0.0634
GRU4Rec	0.0661	0.0913	0.0501	0.0633
JODIE	0.1259	0.1526	0.1203	0.1499
Ours	**0.1425**	**0.1793**	**0.1352**	**0.1631**

5.3 Ablation Study (RQ2)

To investigate how each component affects the model performance, we conduct the following ablation studies, as shown in Table 4:

Table 4: Ablation study evaluated by Recall@20.

Datasets	Electronics		Music	
Ablations	**Product**	**Query**	**Product**	**Query**
Variants on how to construct input graph:				
w/o attr.	0.1686	0.1037	0.1154	0.1132
w/o query	0.1749	-	0.1335	-
w/o product	-	0.0973	-	0.0943
Variants on subgraph sampler:				
Only k-hop sampler	0.1991	0.1203	0.1363	0.1367
Only PPR sampler	0.2123	0.1381	0.1501	0.1399
Static v.s. dynamic:				
Ours (static)	0.1931	0.1183	0.1299	0.1327
Ours	**0.2120**	**0.1393**	**0.1521**	**0.1408**



(a) Effects of # of layers. **(b) Entity type distribution.**

Figure 6: Retrieval analysis. RETE manages to improve performance by stacking more layers, and it can retrieve much more balanced information via a reasonable number of retrieved entities.

5.3.1 The effects of various types of information. : Our solution constructs a temporal KG to organize multi-relational data for joint product and query prediction. As expected, removing rich attributes causes a performance hit, because they provide reliable relations among products and queries. Furthermore, we can demonstrate the mutual benefit derived from the joint query and product prediction task via the performance degradation after removing product entities and query entities, respectively. Interestingly, removing products can significantly affect query prediction performance. This is because queries are mainly connected by product nodes, and a large ratio of query connections is ignored in this ablation.

5.3.2 The effects of ensemble subgraph samplers. : We propose an ensemble subgraph sampler to retrieve relevant entities and filter unrelated noise from the whole graph. Different samplers can capture data characteristics from different perspectives. Only using the k-hop sampler or PPR sampler can hurt the performance compared with ensembling them together. The PPR sampler behaves better than the k-hop sampler, as PPR value can better reflect the relevance among entities when raw neighbor information.

5.3.3 The effects of temporal module. : To evaluate the impact of the temporal attention module, we compare the performance of our static variant. Our dynamic model can have $\sim$ 10% relative improvements over the static variant.

5.4 Analysis of ensemble sampler (RQ3)

We analyze how our ensemble subgraph sampler can improve retrieval results by collecting related higher-order entities and filtering out a large ratio of noises. As shown in Figure 6a, unlike KGAT,

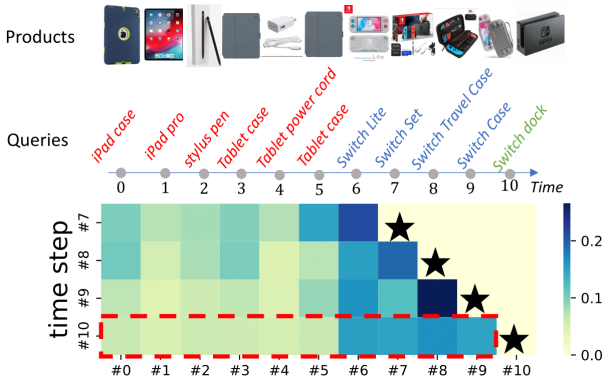


Figure 7: Case study of temporal attention. To forecast the new event at $t = 10$, it emphasizes more on related events, and less on unrelated events. At each time step, we report and summarize the most frequent event.

RETE manages to improve performance by stacking more layers (by default, we choose 3. To investigate the quality of the retrieved entities, Figure 6b shows the distributions among entities types as well as average amount of the retrieved entities. RETE can integrate diverse and balanced information via retrieving a reasonable amount of entities. In contrast, KGAT with 3 layers integrates noisy information from over 9000 entities for each user, where over 96.4% integrated entities are products.

5.5 Temporal Analysis and Case Study (RQ4)

To investigate the evolution of user preference and how RETE can capture it, we select one user from the Electronics dataset with 76 event records. As shown in Figure 7, from the most frequent event in each time step, we can observe an obvious interest shift from mobile tablet-related items to Nintendo Switch-related items. Weights before star are used for auto-regressive forecasting. The temporal module can emphasize more on related events and less on unrelated events at new time steps.

6 CONCLUSION

In this paper, we explore temporal event forecasting, a new problem considering the temporal influence from both query and product to user behaviors. To enhance the sparse action information of most users and meanwhile capture the evolution of user intents, we propose a novel RETE framework to efficiently retrieve similar entities as subgraphs to enrich the user profile representation and then auto-regressively adapt it to be time-aware. We evaluate the proposed RETE method on both product-centric and query-centric event prediction tasks. Extensive experiments on both public and industrial datasets quantitatively and qualitatively demonstrate the effectiveness of the proposed method.

ACKNOWLEDGMENTS

Research reported in this paper was sponsored in part by DARPA award W911NF-17-C-0099, DARPA award HR001121C0165, Basic Research Office award HQ00342110002, and the Army Research Laboratory under Cooperative Agreement W911NF-17-20196.

REFERENCES

[1] Ibrahim Adepoju Adeyanju, Dawei Song, M-Dyaa Albakour, Udo Kruschwitz, Anne De Roeck, and Maria Fasli. Adaptation of the concept hierarchy model with search logs for query recommendation on intranets. In *SIGIR '12*, 2012.

[2] Bijaya Adhikari, Parikshit Sondhi, Wenke Zhang, Mohit Sharma, and B. Aditya Prakash. Mining e-commerce query relations using customer interaction networks. In *WWW '18*, 2018.

[3] Wasi Uddin Ahmad, Kai-Wei Chang, and Hongning Wang. Context attentive document ranking and query suggestion. In *SIGIR'19*, 2019.

[4] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. Learning heterogeneous knowledge base embeddings for explainable recommendation. *CoRR*, abs/1805.03352, 2018.

[5] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '06, page 475–486, 2006.

[6] Ioannis Antonellis, Hector Garcia Molina, and Chi Chao Chang. Simrank++: Query rewriting through link analysis of the click graph. *Proc. VLDB Endow.*, 2008.

[7] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate pagerank. In *KDD'20*, 2020.

[8] Diego Ceccarelli, Sergiu Gordea, Claudio Lucchese, Franco Maria Nardini, and Raffale Perego. When entities meet query recommender systems: Semantic search shortcuts. In *SAC '13*, 2013.

[9] Hanjun Dai, Yichen Wang, Rakshit Trivedi, and Le Song. Recurrent coevolutionary latent feature processes for continuous-time recommendation. In *DLRS 2016*, 2016.

[10] Songgaojun Deng, Huzefa Rangwala, and Yue Ning. Learning dynamic context graphs for predicting social events. In *KDD '19*, 2019.

[11] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. Sequential user-based recurrent neural network recommendations. In *RecSys '17*, 2017.

[12] Albrecht Enders, Harald Hungenberg, Hans-Peter Denker, and Sebastian Mauch. The long tail of social networking. *European Management Journal*, 2008.

[13] Junheng Hao, Tong Zhao, Jin Li, Xin Luna Dong, Christos Faloutsos, Yizhou Sun, and Wei Wang. P-companion: A principled framework for diversified complementary product recommendation. In *CIKM '20*, 2020.

[14] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *SIGIR '17*, 2017.

[15] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR '20*, 2020.

[16] Yunlong He, Jiliang Tang, Hua Ouyang, Changsung Kang, Dawei Yin, and Yi Chang. Learning to rewrite queries. In *CIKM '16*, 2016.

[17] Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. *CoRR*, abs/1706.03847, 2017.

[18] Glen Jeh and Jennifer Widom. Scaling personalized web search. In *WWW '03*, 2003.

[19] Baoyu Jing, Chanyoung Park, and Hanghang Tong. Hdmi: High-order deep multiplex infomax. In *Proceedings of the Web Conference 2021*, pages 2414–2424, 2021.

[20] Baoyu Jing, Hanghang Tong, and Yada Zhu. Network of tensor time series. In *Proceedings of the Web Conference 2021*, pages 2425–2437, 2021.

[21] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR '17*, 2017.

[22] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR'19*, 2019.

[23] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *KDD '19*, 2019.

[24] Jianxun Lian, Xiaohua Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guang zhong Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *KDD'18*, 2018.

[25] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI'15*, 2015.

[26] Fan Liu, Zhiyong Cheng, Lei Zhu, Zan Gao, and Liqiang Nie. Interest-aware message-passing gcn for recommendation. In *Proceedings of the Web Conference 2021*, 2021.

[27] Bhaskar Mitra and Nick Craswell. Query auto-completion for rare prefixes. In *CIKM '15*, 2015.

[28] Dae Hoon Park and Rikio Chiba. A neural language model for query auto-completion. In *SIGIR '17*, 2017.

[29] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD '14*, 2014.

[30] Jiarui Qin, Kan Ren, Yuchen Fang, Weinan Zhang, and Yong Yu. Sequential recommendation with dual side neighbor-based collaborative relation modeling. In *WSDM '20*, 2020.

[31] Lakshmanan Rakkappan and Vaibhav Rajan. Context-aware sequential recommendations withstacked recurrent neural networks. In *WWW '19*, 2019.

[32] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *SIGIR '11*, 2011.

[33] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *WSDM'20*, 2020.

[34] Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R. Voss, and Jiawei Han. Automated phrase mining from massive text corpora. In *SIGMOD'15*, 2017.

[35] H. Shao, S. Yao, A. Jing, S. Liu, D. Liu, T. Wang, J. Li, C. Yang, R. Wang, and T. Abdelzaher. Misinformation detection and adversarial attack cost analysis in directional social networks. In *ICCCN'20*, 2020.

[36] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. Session-based social recommendation via dynamic graph attention networks. In *WSDM '19*, 2019.

[37] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM '19*, 2019.

[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR'17*, 2017.

[39] Haiwen Wang, Ruijie Wang, Chuan Wen, Shuhao Li, Yuting Jia, Weinan Zhang, and Xinbing Wang. Author name disambiguation on heterogeneous information network with adversarial representation learning. In *AAAI '20*, 2020.

[40] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *CIKM '18*, 2018.

[41] Ruijie Wang, Zijie Huang, Shengzhong Liu, Huajie Shao, Dongxin Liu, Jinyang Li, Tianshi Wang, Dachun Sun, Shuochao Yao, and Tarek Abdelzaher. Dydiff-vae: A dynamic variational framework for information diffusion prediction. In *SIGIR'21*, 2021.

[42] Ruijie Wang, Yuchen Yan, Jialu Wang, Yuting Jia, Ye Zhang, Weinan Zhang, and Xinbing Wang. Acekg: A large-scale knowledge graph for academic data mining. In *CIKM '18*, 2018.

[43] Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, and Quoc Viet Hung Nguyen. Streaming ranking based recommender systems. In *SIGIR '18*, 2018.

[44] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In *KDD '19*, 2019.

[45] Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI'11*, 2011.

[46] Yuchen Yan, Lihui Liu, Yikun Ban, Baoyu Jing, and Hanghang Tong. Dynamic knowledge graph alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021.

[47] Yuchen Yan, Si Zhang, and Hanghang Tong. Bright: A bridging algorithm for network alignment. In *Proceedings of the Web Conference 2021*, 2021.

[48] Chaoqi Yang, Jinyang Li, Ruijie Wang, Shuochao Yao, Huajie Shao, Dongxin Liu, Shengzhong Liu, Tianshi Wang, and Tarek F. Abdelzaher. Hierarchical overlapping belief estimation by structured matrix factorization. In *ASONAM'20*, 2020.

[49] Chaoqi Yang, Ruijie Wang, Shuochao Yao, and Tarek F. Abdelzaher. Hypergraph learning with line expansion. *CoRR*, abs/2005.04843, 2020.

[50] Chaoqi Yang, Ruijie Wang, Shuochao Yao, Shengzhong Liu, and Tarek F. Abdelzaher. Revisiting "over-smoothing" in deep gcns. *CoRR*, abs/2003.13663, 2020.

[51] Hanqing Zeng, Muhan Zhang, Yinglong Xia, Ajitesh Srivastava, Andrey Malevich, Rajgopal Kannan, Viktor K. Prasanna, Long Jin, and Ren Chen. Deep graph neural networks with shallow subgraph samplers. In *NeurIPS'21*, 2021.

[52] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *KDD '16*, 2016.

[53] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. How to retrain recommender system? a sequential meta-learning method. In *SIGIR '20*, 2020.

[54] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in {gnn}s. In *International Conference on Learning Representations*, 2020.

[55] Sijin Zhou, Xinyi Dai, Haokun Chen, Weinan Zhang, Kan Ren, Ruiming Tang, Xiuqiang He, and Yong Yu. Interactive recommender system via knowledge graph-enhanced reinforcement learning. In *SIGIR'20*, 2020.

A APPENDIX

A.1 Theoretical analysis

Section 4.1 specifies two model requirements for temporal event forecasting task. In this section, we analysis how RETE satisfies them theoretically.

- *Requirement 1*: It should learn informative and discriminative user representations by considering higher-order information and filtering out large ratio of noise from whole graph.
- *Requirement 2*: It should capture user intent evolution from data collected at different time steps, so as to produce up-to-date forecasting.

For *Requirement 2*, it is straightforward to show that our temporal attention module can auto-regressively consider new data and update the most up-to-date intents. To analysis *Requirement 1*, we propose the following proposition:

Proposition A.1. *For temporal event forecasting task, models able to learn informative and discriminative user representations should satisfy the following necessary conditions: let $m_{[u]}$ denote information gained by user u via n-layer of neighbor aggregation, gained information of different users should always be distinguishable: $m_{[u]} \neq m_{[v]}, (u \neq v)$, even under $n \to \infty$.*

The correctness of proposition A.1 is obvious, as models may utilize a large number of layers of neighbor aggregation to integrate higher-order and related information, especially for users with sparse records. So it should be satisfied under n up to ∞. To analyze why RETE satisfies the this condition, let $m_{[u]} = \{m^1_{[u]}, m^2_{[u]}, ..., m^t_{[u]}\}$ denote gained information by user u from different time steps. By applying theorem in [51], gained information at time step t after ∞-layer of GCN-like neighbor aggregation (used in our structural attention module) can be represented as:

$$m^t_{[u]} = \sqrt{\frac{\delta_{[u]}(u)}{\sum_{v \in \mathcal{G}^t_{[u]}} \delta_{[u]}(v)}} \cdot \boldsymbol{e}^T_{[u]} \boldsymbol{X}_{[u]} \tag{10}$$

where $\delta_{[u]}(v)$ denotes degree of node v in sampled subgraph $\mathcal{G}^t_{[u]}$, $\boldsymbol{X}_{[u]}$ denotes initial entity embeddings in sampled subgraph $\mathcal{G}^t_{[u]}$, and $\boldsymbol{e}_{[u]}$ denotes eigenvector of $\mathcal{G}^t_{[u]}$ corresponding to largest eigenvalue. By using ensemble subgraph sampler, we can ensure $\mathcal{G}^t_{[u]} \neq \mathcal{G}^t_{[v]}$ even after ∞-layer of neighbor aggregation, so that $m^t_{[u]} \neq m^t_{[v]}$ in each time step, and $m_{[u]} \neq m_{[v]}$. Without such subgraph constrains, after large number of neighbor aggregations, the integrated entities easily span the whole graph, i.e., $\mathcal{G}^t_{[u]} = \mathcal{G}^t_{[v]} = \mathcal{G}^t$, making learned user representations much less distinguishable.

A.2 TransR for product graph learning.

Rich meta-data of products forms a heterogeneous product graph $\mathcal{G}_P$, describing important attributes of each product $p \in \mathcal{P}$. Specifically, $\mathcal{G}_P = \{(e, r, e') | e \in \mathcal{P}, e' \in \mathcal{I} \bigcup \mathcal{Q}, r \in \mathcal{R}_P\}$, where $\mathcal{I}$ denotes attribute set for products, including but not limited to brand, product type and category. $\mathcal{R}_P$ denotes the relation set among them. Each triple $(e, r, e') \in \mathcal{G}_P$ represents a fact indicating that product entity e associate with tail entity e' through relation r. $\mathcal{G}_P$ also describe *mapping* relations between products and queries.

To better represent product and query information from static product graph, we utilize a knowledge graph embedding module TransR [25] to project them into the same low-dimensional space. The objective is shown below:

$$\mathcal{L}_{KGC} = \sum_{(e,r,e') \in \mathcal{G}_P} \sum_{(e,r,e^-) \in \mathcal{G}_P^-} \max\left(0, f_r(e, e') + \lambda_{KGC} - f_r\left(e, e^-\right)\right) \tag{11}$$

where $\mathcal{G}_P$ denotes product graph, $\mathcal{G}_P^-$ denotes negative samples, λ_{KGC} is the margin value. Following TransR, we define $f_r(h, t) = \|\mathbf{h}\mathbf{W}_r + \mathbf{r} - \mathbf{t}\mathbf{W}_r\|_2^2$, where $\mathbf{W}_r$ is trainable relation matrix.

A.3 Implementation

We implement RETE with Python 3.8.5. We use PyTorch 1.9.1 on CUDA 11.1 to train RETE on GPU. To implement fast subgraph sampling, we adopt methods in [5, 51] to calculate the approximate PPR value by only traversing the local region around each user. For better efficiency, we implement the sampling part with C++ and the interface between C++ and Python is via PyBind11.

A.4 Experimental setup

A.4.1 Data collection. **Yelp**. The dataset is adopted in Yelp Challenge 2019 [3], which contains the interaction records between users and businesses like restaurants and bars. We utilize review history between users and businesses as user-product interactions. And we extract discriminative keyphrases from reviews as queries, so as to collect pseudo user-query interactions. We are able to show that jointly consider both pseudo queries and products can improve both performances. We propose the following procedure to collect experimental data.

(1) We collect data from April 2014 to Jan 2021 and preserve those with ratings higher than 3.0, as high ratings indicate true user intents.
(2) To extract keyphrases as pseudo query, we utilize both AutoPhrase [34], 2-gram, and 3-gram methods to extract keyphrases. Then we calculate TF-IDF scores for each and preserve top 15000 as pseudo query pools.
(3) We adopt 20-core setting to collect entities, i.e., we remove users/products/queries with fewer interactions than 20.
(4) We divide the time span into 28 time steps according to interaction timestamps. We split them into background/training/val/test (10/10/2/6).
(5) To construct product graph, we first preserve critical attributes like category, location, etc to construct product-to-attribute edge, we collect frequent pairs of businesses from background data as product-to-product edge, we collect query-to-product edge also from background data per each review action.

E-commerce We gain access to the search log data including 140 days and product attribute data. We first collect data under four specific categories: *Electronics*, *Book*, *Music* and *Beauty*. For each category, We propose the following procedure to collect experimental data.

(1) We collect data from Feb 2021 to June 2021 and preserve those with specific types of actions: click, add cart, follow-on click, and purchase, with ratios of all data 11.8%, 5.1%, 4.6% and 2.0% respectively. They are preserved as they reveal strong signals

[3] https://www.yelp.com/dataset/

Table 5: Statistics of experimental datasets.

	Public	Industrial (E-commerce)			
Dataset	**Yelp**	**Electronics**	**Music**	**Book**	**Beauty**
#User	22,307	5,928	7,453	37,562	47,261
#Product	16,153	10,129	12,105	61,215	51,686
#Query	9,314	8,045	6,506	25,340	25,807
Product interactions	820,219	138,607	582,651	2,976,112	1,385,366
Query interactions	800,727	58,037	213,281	718,035	200,219
#Entity	49,269	29,212	28,643	134,370	140,314
#Triplet	1,791,788	496,701	1,832,501	7,739,316	3,092,010
Time span	80 months	140 days	140 days	140 days	140 days
#Time step	28	28	28	28	28

of user intents. For each action, we record user, query/product, timestamp, and action type.

(2) We adopt 10-core setting to collect entities, i.e., we remove users/products/queries with fewer interactions than 10.

(3) We divide the time span into 28 time steps according to interaction timestamps. We split them into background/training/val/test (10/10/2/6).

(4) To construct product graph, we first preserve critical attributes like brand, model, etc to construct product-to-attribute edge, we collect frequent pairs of products within the same sessions as product-to-product edge, we collect query-to-product edge also from background data if users have interaction to one product via one search query.

We purposefully choose the two platforms because of the various time period. The e-commerce dataset emphasizes more on user short-term interest since the shopping intent is more time-sensitive. By contrast, the Yelp dataset emphasizes more on user long-term interest, since it lasts longer and a user's choice on businesses is less time-sensitive.To evaluate our framework, we divide the time span into 28 time steps according to interaction timestamps. We split them into background/training/val/test (10/10/2/6) to train initial entities embeddings (model input), train, validate and test our model respectively. Table 5 summarizes the statistics of the experimental datasets.

A.4.2 Baseline.

- **FM** [32]. This is a basic factorization model which considers the second-order connections. We construct input features as multi-hot vectors.
- **NFM** [14]. This is a state-of-the-art factorization model, which subsumes FM under neural networks. Specially, we employed one hidden layer to extract features from inputs.
- **GRU4Rec** [17]. This is a sequential recommendation method that utilizes gated recurrent units (GRU) to learn temporal information of user action sequences.
- **BERT4Rec** [37]. This is a sequential recommendation method that utilizes BERT to learn temporal information of user action sequences.
- **JODIE** [23]. This is a dynamic graph method that considers co-evolution of both users and products. We encode rich side information for it via initial embedding.
- **ECFKG** [4]. This is an advanced collaborative filtering framework that incorporates knowledge graph for recommendation and utilizes KG embedding to learn representations.
- **KGAT** [44]. This is an end-to-end knowledge graph attention network that explicitly models the high-order connections and heterogeneous entities and employs an attention mechanism to discriminate the importance of the neighbors.

A.4.3 Setup. It is worth noting that all baselines are designed for recommending products. To generalize them to be able to predict queries, we train them using product and query loss separately. Those (KGAT, ECFKG) that consider knowledge graph can explicitly fuse information from both product and query. For all methods that need initialization, we utilize a classic and lightweight knowledge embedding method, TransR, to represent multi-relational information in background data, not just IDs. For dynamic methods (BERT4Rec, GRU4Rec, JODIE) that aim to only predict the next interacted product/query, we modify the evaluation protocol to predict all possible products/queries in the following time steps.

For fair comparison, we do not utilize rich semantic information from query entities and product descriptions, as all compared baselines only consider interaction records and structural side information. We fix the dimension of latent vectors of all methods as 128, and we report the average performance of the best model on the validation set. For RETE, we tune learning rate within $\{0.0001, 0.0005, 0.001, 0.005, 0.01\}$ and regularization weight $\{0.005, 0.05, 0.5\}$ according to *Recall*@20 of product prediction on validation set. We ensemble one PPR sampler and one randomized 3-hop sampler to retrieve subgraphs, and we stack 3 layers of graph attentions to better integrate information.