Prompt Learning Unlocked for App Promotion in the Wild

Anonymous Author(s)
Affiliation
Address
email

Abstract

In recent times, mobile apps have increasingly incorporated app promotion ads to promote other apps, raising cybersecurity and online commerce concerns related to societal trust and recommendation systems. To effectively discover the intricate nature of the app promotion graph data, we center around the graph completion task, aiming to learn the connection patterns among diverse relations and entities. However, accurately deciphering the connection patterns in such a large and diverse graph presents significant challenges for deep learning models. To overcome these challenges, we introduce `Prompt Promotion`, a transformer-based framework that unlocks *prompt learning* capabilities by incorporating metapath- and embedding-based prompts that provide valuable hints to guide the model's predictions for undetermined connection patterns. Experimental results show that our `Prompt Promotion` model represents a pioneering prompt-based capability in effectively completing the app promotion graph. It not only demonstrates superior performance in heterogeneous graph completion in real-world scenarios, but also exhibits strong generalization capabilities for diverse, complex, and noisy connection patterns when paired with their respective prompts.

1 Introduction

Figure 1: Example of malicious app promotion.

Mobile applications, or apps, often incorporate advertisements (ads) as a means of promotion (Viennot et al., 2014; Liu et al., 2015), among which app-promotion ads are commonly used by Android app developers to promote other apps (Research, 2023). However, concerns arise regarding the trustfulness of the apps promoted through these ads, given the competitive nature of the industry and the potential for the promotion of malicious apps (Rafieian and Yoganarasimhan, 2021; Son et al., 2017; Hardt and Nath, 2012). Previous research has focused on analyzing the behaviors of ad libraries within the app promotion ecosystem (Grace et al., 2012; Vallina-Rodriguez et al., 2012; Nath, 2015; Jin et al., 2021; Liu et al., 2020). However, these studies primarily examine the behaviors of ad libraries themselves, and pay too little attention to app propagation in terms of how massive individuals exploit the app promotion ecosystem. For instance, Figure 1 illustrates an app promotion chain where a popular benign app "Passport Photo Maker - ID/VISA" promotes a greyware app "Photo Collage, Photo Editor", which in turn promotes malware "Flood-It!", a strategy game capable of scanning the local network and stealing sensitive phone information. Furthermore, these studies lack a comprehensive understanding of app promotions, which involve multiple heterogeneous actors beyond apps, such as app markets, security vendors, and developers.

For example, Figure 2 provides an inference path to explain why an online messaging app, "Polish English Translation", promotes "CallApp". The underlying behaviors indicate that "Polish English Translation" shares the same developer as another translation app "Thai Chinese Translation", which has been observed to promote "CallApp". Hence, a more holistic approach that learns the intrinsic connection patterns among these various entities is necessary to deeply understand the complexities of the whole app promotion ecosystem and its implications for society and online commerce. To address these limitations, we employ insights of graph completion learning into the heterogeneous app promotion graph. Our goal is to predict unknown target entities based on known source entities and relation queries, thereby completing the full graph. By applying graph completion methods to the app promotion graph, we are able to learn representations that capture the intricate connection patterns among different types of entities and relations. This approach not only sheds light on the underlying dynamics of app promotion graphs, but also opens up possibilities for diverse applications.

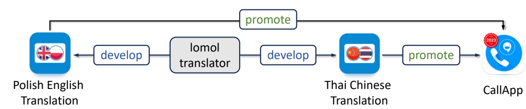


Figure 2: Example of inferred app promotion path.

Nevertheless, learning to complete the focused app promotion network is non-trivial, especially with datasets collected from the wild. Existing methods for graph completion are either too simplistic for modeling the network complexity and information among relationships and entities (Bordes et al., 2013; Sun et al., 2018; Yang et al., 2015), or they heavily rely on rich semantic information to train massive weight parameters (Wang et al., 2021; Lv et al., 2022; Yao et al., 2019), which contradicts the scarcity of semantic information in app promotion networks collected from the wild. Therefore, in this work, considering the challenge of modeling complex connection patterns while overcoming the limitations of existing techniques, we introduce our approach `Prompt Promotion`, which guides the model in learning the intricate connection patterns by incorporating a combination of embedding-based and metapath-based prompts. Leveraging the power of pretrained BERT, we design the embedding-based prompts, derived from pretrained embedding-based methods like DistMult (Yang et al., 2015), provide prior knowledge as hints to assist the model in making informed references. Additionally, we further craft metapath-based prompts by extracting not only valid but also informative metapaths for each queried relation. Subsequently, we combine the embedding-based and metapath-based prompts along with the query tokens, and randomly permute them to form the final input sequence for each query. The sequence is tokenized using the embedding-based method, replacing the original BERT tokenizer, to ensure that the tokens are projected into the same embedding space for the subsequent fine-tuning process. In summary, the contributions of this paper are:

- We propose a novel approach named `Prompt Promotion`, that addresses the challenge of modeling connection patterns in complex app promotion graphs by leveraging the pretrained BERT as the backbone model while incorporating both the embedding-based and metapath-based prompts to guide the model in learning the intricate patterns within the graph.
- We demonstrate the effectiveness of our approach through extensive experiments on our collected real-world dataset. The results show that our approach outperforms existing techniques in terms of accuracy and generalization capabilities in extracting diverse and complex connection patterns.
- We contribute to the research community by providing a deeper understanding of the app promotion ecosystem, its complexities, and implications for societal trust and online commerce. Our work sheds light on the potential applications of graph completion methods, specifically utilizing pretrained BERT, in improving trustworthiness in detecting malicious apps.

2 Background

In this section, we briefly introduce the definitions of a heterogeneous graph, metapath, and the task of heterogeneous graph completion, as well as the idea of prompt engineering and details about our app promotion graph dataset. We additionally refer related works in Appendix E.

2.1 Definitions

Definition 1 (Heterogeneous Graph)**.** *A heterogeneous graph (HG) $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ consists of a node set $\mathcal{V}$, an edge set $\mathcal{E}$, and the optional features of the associated nodes and edges: $\mathcal{X} = (\mathcal{X}_{\mathcal{V}}, \mathcal{X}_{\mathcal{E}})$.*

Each node's type is mapped through the node type mapping function $\phi : \mathcal{V} \to \mathcal{A}$, and each edge's type through the edge type mapping function $\psi : \mathcal{E} \to \mathcal{R}$, where $\mathcal{A}$ and $\mathcal{R}$ denotes the node and edge type set respectively. We represent each edge as a triple $(h, r, t) \in \mathcal{E}$ where $h, t \in \mathcal{V}, r \in \mathcal{R}$, and the edges are directional. For a heterogeneous graph, there exists the constrain $|\mathcal{A}| + |\mathcal{R}| > 2$.

Definition 2 (Metapath)**.** *In a heterogeneous graph, a metapath represents a predefined sequence of node types and edge types that capture the desired semantic relationships between nodes. Formally, a metapath $\mathcal{P}$ is denoted as $e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} ...e_L \xrightarrow{r_L} e_{L+1}$, where $r_i \in \mathcal{R}, e_i \in \mathcal{A}, r = r_1 \cdot r_2 \cdot ... \cdot r_L$ is the composite relation between entity type e_1 and e_{L+1}, and L is the length of the metapath.*

Definition 3 (Heterogeneous Graph Completion)**.** *For a valid query (h, r) where $h \in \mathcal{V}$ and $r \in \mathcal{R}$, a heterogeneous graph completion (HGC) task refers to discovering valid answers $\mathcal{T} \subset \mathcal{V}$ such that for all $t \in \mathcal{T}$, $(h, r, t) \in \mathcal{E}$.*

2.2 Prompt Engineering

Prompt engineering is a systematic methodology widely employed in natural language processing applications to craft specific input signals to invoke desired output responses from machine learning models. Under the task of graph completion where input tokens are the queries, prompts can be designed as contextual semantic information related to the entities and relations, the query-related neighborhood, or other encoded information that guides the model to answer the query. Specifically for a query (h, r) in the graph completion task, the prompted input sequence is usually formulated as:

```
[<bos>] <prompts> [<sep>] <h> <r> [<sep>].
```

2.3 App Promotion Dataset

2.3.1 Data Collection

The dataset pertaining to app promotion is gathered from three distinct perspectives. Initially, for each app, the package name, developer information, and category of each app are crawled from Google Play. Subsequently, an analysis is conducted on the app using VirusTotal to examine the flags associated with its security level, along with the corresponding URLs. Lastly, the manifest and signature of each app are inferred through the process of reverse engineering (e.g., interesting strings provided by the VirusTotal report). The promotion actions between apps are discovered by checking whether the clickable widgets in a UI from the source apps lead to the download page of the sink app. If so, then a source_app <promotes> sink_app relation is identified. The collected raw data is then used to construct the following HG for the capture of ample behavior patterns.

2.3.2 Graph Construction

In order to harness the informative attributes of applications, such as URLs and signatures, which are instrumental in forecasting elusive promotional strategies and discerning recurrent patterns in app promotion, we construct the App Promotion HG (APHG) to epitomize the sundry entities and relations inherent in the network. More details related to the entity statistics and relations of constructed graph are provided in Appendix A.

Entities. An APHG encapsulates distinct entities derived from the following app attributes: application package name, developer, application category, manifest, VirusTotal (VT) Engine, digital signature, and URL. The manifest entity encompasses app activities, providers, receivers, services, and permissions. Given the unique promotional behaviors demonstrated by benign, greyware, and malicious applications, we further classify the application package name into these three discrete classes, and extend the aggregate count of entity types within our framework to nine.

Relations. We consider multiple directional relations among the entities defined above to capture their interactive behaviors: *app-promote-app*, *app-include-signature*, *engine-detect-app*, *app-belong-category*, *developer-involve-category*, *developer-develop-app*, *app-access-URL*, *developer-use-URL*, *app-own-manifest*. Since apps with different security levels follow different behavior patterns, we further divide them into three sub-classes: *benign*, *grey* and *malicious*. In total, the above relations are extended to twenty-nine classes of relation types. Note that all the relations are directional, and each query only associates with one of the constructed directional relations, excluding the reverse relations. Despite the potential to gather additional information, neither the entities nor the relations are associated with any features. Therefore, our APHG is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

2.3.3 Task Motivation

Despite the relations in place abundantly capturing intrinsic application information, instances of information paucity are far from scarce. This scarcity impedes our ability in gathering all relevant information (see an illustration on Google Play in Appendix B), which in turn substantially impedes our capacity to decipher patterns in application promotion behavior. Thus, to alleviate the burden of information scarcity, we propose to first target the HGC task on our app promotion graph.

3 Unlocking Prompt Definition on HGC

3.1 Overall Framework

Our `Prompt Promotion` approach leverages the pre-trained BERT (Devlin et al., 2019) as the transformer encoder to encode the tokenized input sequence of each query. We use an aggregator to consolidate the output sequence and a two-layer MLP as the prediction head to perform the final task. The overall framework is depicted in Figure 3. We incorporate this design for two reasons: (1) encoding the query instead of the triple mitigates the calculation overhead when responding to a specific query, and (2) the attention mechanism in BERT assigns global attention to the provided input, including the designed prompts. We extend the input for each query into three parts: embedding-based prompts, metapath-based prompts, and the query itself, consisting of a source-relation pair. In the following content, we provide details regarding the two sets of prompts.

3.2 Embedding-based Prompts

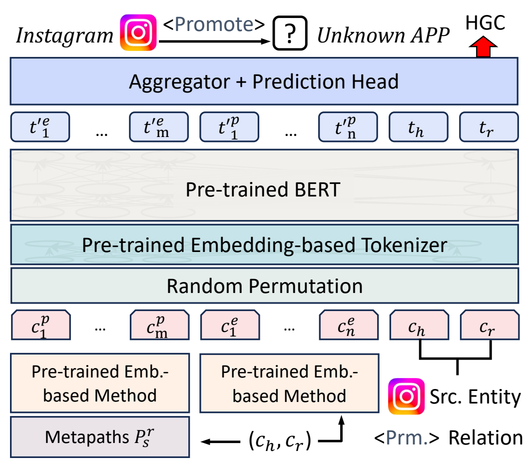


Figure 3: Overall framework of our `Prompt Promotion`.

Prior embedding-based models have demonstrated remarkable performance on various public benchmark datasets, making them state-of-the-art solutions. These models possess inherent simplicity that renders them proficient tokenizers, effectively mapping entity and relation tokens to a shared semantic space. In this paper, we select DistMult (Yang et al., 2015) as the pre-trained embedding-based method to tokenize the entity and relation tokens. Note that this is a designer's choice and can be substituted with any other methods that fit our framework. The n embedding-based prompts are defined as the top-n predicted entities by the pretrained embedding-based methods according to the predicted scores, denoted as C^e. These prompts serve as prior knowledge that assists the model in making informed references. For instance, when considering the query "*which app does Instagram promote?*", we provide additional prompts in the form of a hint, such as "*I am not 100% sure, but I believe these apps might be the answers.*" This supplementary information aids the model in further generating more accurate and contextually relevant responses. The filtered prompted entities are then tokenized with the corresponding embeddings learned by the pre-trained embedding-based method.

3.3 Metapath-based Prompts

While embedding-based prompts serve as hints from the pioneers, they may neglect true answers due to their narrowed perspectives. Typically, embedding-based methods utilize geometric operations in the representation space, resulting in prompts that share similarities from a geometric perspective. Although entities in a knowledge graph inherently possess semantic meanings, we posit that the semantic information of entities in a heterogeneous graph can be alternatively extracted from metapaths. Metapaths offer a means to capture and encode meaningful relationships within the graph, facilitating the extraction of valuable insights. Therefore, we further provide the model with prompts from another perspective, i.e., the metapath-based prompts from the semantic perspective.

The key assumption lies in the connection between a certain metapath and the queried relation. In the following content, we first introduce the measure of the correlation between a metapath and a query, and then illustrate how to utilize the correlation to create the metapath-based prompts.

3.3.1 Query-Metapath Correlation

For a clearer clarification, we first define the functions $src(\cdot)$ and $dst(\cdot)$ as the source and destination entity type extractions for a relation r respectively. Regarding a specific queried relation, we make the following definition:

Definition 4 (r-valid Metapath)**.** *A metapath $p = e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} ...e_L \xrightarrow{r_L} e_{L+1}$ is r-valid if and only if e_1 and e_{L+1} are the source and destination entity types of relation r, respectively.*

For example, for the relation r =*benign-access-URL*, a corresponding valid metapath includes but is not limited to *benign* $\xleftarrow{develop}$ *developer* $\xrightarrow{use}$ *URL*, where $src(r) =$ *benign* and $dst(r) =$ *URL*. The first step of linking a queried relation with a certain metapath is to identify all the r-valid metapaths. For multi-hop reasoning tasks, the answers to a query usually lie within three hops. We control the length of the metapath as $L \leq 2$ and conduct an exhaustive search for each query relation $r \in \mathcal{R}$, where $\mathcal{R}$ denotes the set of all relations. The set of all r-valid metapaths is denoted as $\mathcal{P}_r$. Note that when searching for r-valid metapaths, we also consider the reverse relation of the original relation, since there exist entities with only outgoing edges, and reverse relations do not change the semantic meanings. However, we only consider the original relations as the queried relations. The metapaths in $\mathcal{P}_r$ are valid, but not necessarily informative. In other words, $\mathcal{P}_r$ does not inform us how relevant each $p \in \mathcal{P}_r$ is to r. To quantify the correlation, we make the following definitions:

Definition 5 (p-Hit)**.** *For a specific triple (h, r, t), where r is the relation, h is the source entity such that $h \in \mathcal{H} \subset \mathcal{V}$ and $\phi(h) = src(r)$, t is the destination entity such that $t \in \mathcal{T} \subset \mathcal{V}$ and $\phi(t) = dst(r)$, we say the triple (h, r, t) is p-Hit if and only if there exist at least one path from h to t such that this path is an instance of the metapath p.*

Definition 6 (p-Hit Ratio)**.** *For a specific triple (h, r, t), if this triple is p-Hit, then the p-hit ratio α of this triple is defined as the ratio of t among all other entities reached by the metapath p; otherwise, the p-hit ratio of this triple is zero.*

Definition 7 (r-p Ratio)**.** *For a specific relation r, a metapth $p \in \mathcal{P}_r$, and all true (h, r, t) triples, the corresponding r-p ratio is defined as the averaged hit ratio of all true r related triples, i.e., triples constructed with relation r. Note that the ratio is calculated based on a filtered setting: if t' is a correct answer to the query (h, r) when evaluating on the answer t, we remove t from the denominator.*

We here provide a concrete example for examplification. Consider the relation $benign$-$access$-URL and its valid metapath $p =$ *benign* $\xleftarrow{develop}$ *developer* $\xrightarrow{use}$ *URL*. For each true triple $(h, benign$-$access$-$URL, t)$ such that $\phi(h) = benign$ and $\phi(t) = URL$, denote the set of accessible URLs to the query $(h, benign$-$access$-$URL)$ as $\mathcal{T}_h$, and the set of all URL entities reached by following metapath p starting from h as $\mathcal{T}_h^p$. If the triple is p-Hit, then the hit ratio is calculated as $\alpha = 1/(|\mathcal{T}_h^p \setminus \mathcal{T}_h| - 1)$; otherwise, $\alpha = 0$. We minus one in the denominator because $t \in \mathcal{T}_h$. The r-p ratio of the relation $benign$-$access$-URL is then calculated as the averaged α of all the related true triples. Naturally, if a metapath p is highly correlated with r for a specific source entity h, the corresponding α should be high. We utilize the r-p ratio of each relation-metapath pair as the correlation indicator to select the top-m metapaths for further prompt generation, and denote the m selected metapaths as $\mathcal{P}_r^s$.

3.3.2 Metapath-based Prompt Generation

Even though we select m metapaths for each query, some metapaths may contain noise. This is especially true when a metapath reaches a high-degree entity, resulting in a significant expansion of the candidate pool. In such cases, these prompts may not provide any substantial additional information beyond what is already known, rendering them less informative. To address this, we apply a candidate filtering method. Specifically, we utilize a limit l to separate metapaths that lead to large or small candidate sizes. For small-sized candidates, we perform the *union* operation, and for large-sized candidates, we perform the *intersect* operation. The rationale is as follows: some queries may not be highly relevant to just one metapath, in which case the number of candidates is usually large, and we rely on the *intersect* operation to filter out noise. On the other hand, some queries may be explained by more than one metapath, in which case the size of the candidate pool is usually small, and the *union* operation considers all conditions.

After the filtering process, we empirically evaluate the correlation between one queried relation and the filtered candidates. Particularly, We calculate the average size of the filtered candidate pools s_r for all r related triples, as well as the hit ratio h_r of the correct answer for each type of query among the candidate pools. In addition, we denote the base hit ratio as $b_r = s_r/|\phi(t)|$ and the magnification as $m_r = h_r/b_r$. Table 1 presents a selection of the evaluation results due to the large size of $\mathcal{R}$. The table provides rich information: (1) the selected metapaths for some relations are highly correlated with their relations, indicated by high h_r and low s_r (e.g., `mal-belong-category`); (2) some other relations provide a considerable amount of correlation, indicated by a large m_r, but may lead to a high hit ratio (e.g., `benign-access-URL`) or a low hit ratio (e.g., `developer-use-URL`), affected by s_r; (3) there are also cases in the middle with decent h_r and s_r (e.g., `grey-promote-grey`). Nevertheless, the results confirm that metapaths provide information regarding the query, regardless of high or low h_r. To reduce the size of the input prompts, we further utilize an embedding-based method to select the top-m prompts among the candidate set $\mathcal{C}_h^r$ as the final metapath-based prompts, denoted as $\mathcal{C}^p$.

Table 1: Empirical evaluation results of the correlation between metapath and relation.

Relation	h_r	s_r	b_r	m_r
`mal-belong-category`	0.922	4.932	0.137	6.732
`benign-access-URL`	0.879	129.329	0.007	128.1
`developer-use-URL`	0.457	42.905	0.002	200.486
`grey-promote-grey`	0.660	154.786	0.135	4.876

3.4 Combined Input Sequence

For a query (h, r), we concatenate the embedding-based prompts C^e, the metapath-based prompts C^p, and the query token h and r as the final input sequence. Before feeding the constructed sequence into the pre-trained BERT model, we randomly permute the tokens. This step is essential in forcing the BERT model to learn the intrinsic connection between the query and the answer, rather than relying too much on the prompts. We validate the necessity of this step in the following experiments. After the permutation, the input sequence is tokenized via the embedding-based method, replacing the original BERT tokenizer. Finally, we adopt the binary cross entropy loss for the HGC task. We provide the pseudo code for our method in Appendix C.

4 Experiment

4.1 Setup

We test our method's effectiveness over the constructed APHG as decribed in Section 2.3. For comparison, we carefully select DistMult (Yang et al., 2015), ComplEX (Trouillon et al., 2016), ConvE (Dettmers et al., 2018), HittER (Chen et al., 2021), and LTE (Zhang et al., 2022) as the baselines, for they can be easily adapted to our HGC task. For evaluation purposes, we adopt two key metrics: mean reciprocal rank (MRR) and Hits@K, and higher values of MRR and Hits@K indicate better performance in accurately ranking and identifying the correct candidates in the graph completion task. We use a pre-trained DistMult (Yang et al., 2015) as the backbone model to tokenize the entities and relations as low-dimensional vectors, and utilize a pre-trained ComplEX (Trouillon et al., 2016) for prompt filtering. Note that these choices are a matter of preference, and can be substituted with other embedding-based methods such as TransE (Bordes et al., 2013). We consider two settings under our framework: *w/ Rand. Perm.* denotes that we randomly permute the input tokens before the encoding process, and *w/o Rand. Perm.* suggests otherwise. The input sequence is decomposed into three essential components - the embedding-based prompts, metapath-based prompts, and the query. Based on the above settings and components, we define model variants as shown in Table 2. More detailed experimental setups are provided in Appendix D due to space limit.

4.2 Performance on App Promption

The performance comparison in Table 3 demonstrates that our model outperforms the other baselines by a significant margin. This improvement can be attributed to two key factors: the incorporation of the designed prompts and the utilization of random permutation. While our model utilizes

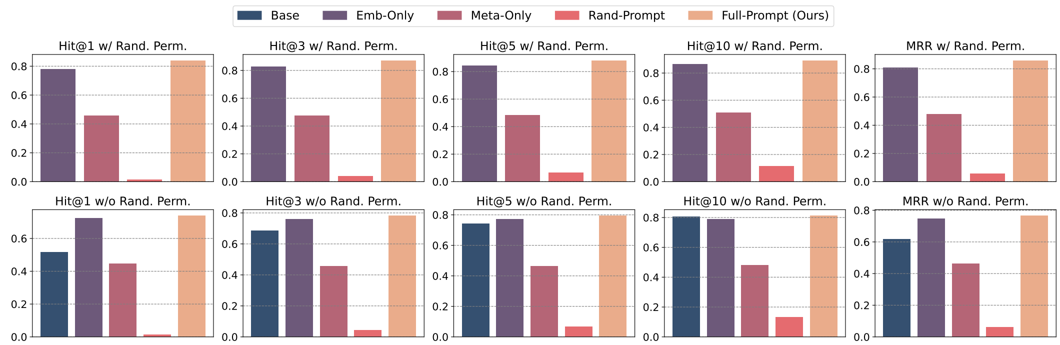


Figure 4: Results of component-differed variants, including ours (Full-Prompt w/ Rand. Perm.).

DistMult (Yang et al., 2015) as the backbone, it extends its capabilities beyond a simple multiplication projection of the queried source entity and relation embeddings. This is evident from the consistent notable performance enhancement achieved by our model. We also observe that as the value of K increases, the performance gap between our model and the baselines gradually diminishes. We hypothesize that our model follows a two-step inference process: first, it processes the provided prompts and attempts to identify potential answers out of the input sequence. If the correct answers are present in the prompts, the model can recognize them with relatively high probabilities, leading to higher hit ratios when K is small. This aspect of the task is relatively straightforward. However, if the answers are not found in the provided prompts, the model transits to another task and endeavors to generate an answer by considering all the given hints. This second task tests the model's ability to deduce query patterns and is inherently more challenging. We refer to this hypothesis as the "dual-task" hypothesis, which suggests that our model performs and excels at both the answer identification and answer generation tasks. Additionally, we observe a notable performance downgrade among all the variants compared to the best. Under most conditions, the BERT encoder significantly improves Hit@1 performance, suggesting that our framework focuses more on direct query answering, rather than pattern matching. We provide more detailed analysis in the following section to validate our "dual-task" hypothesis, and examine the model's capabilities under several conditions.

4.3 Component Analysis for Prompt Designs

In this part, we further analyze the impacts of each component in our framework to confirm the necessity of constructing our model as designed, as well as providing supportive evidence for our "dual-task" hypothesis. We add another variant *Random-Prompt*, where the input sequence is constructed with randomly sampled prompts plus the query tokens.

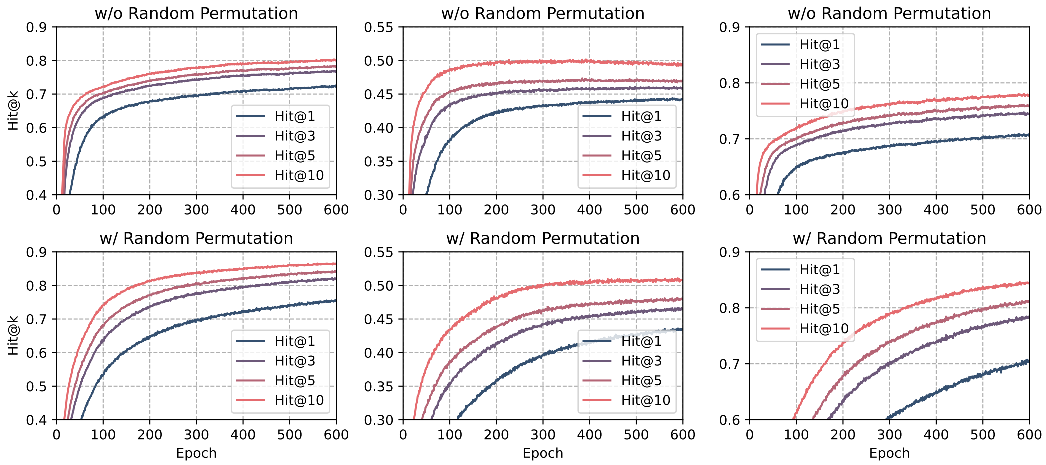
Table 2: Definitions of variants of our `Prompt Promotion`.

Variant	Emb. Prm.	Mtp. Prm.	Query	Rand. Perm.
Base	✗	✗	✓	✓
Emb.-based Only	✓	✗	✓	✓
Mtp.-based Only	✗	✓	✓	✓
Ours w/o Rand. Perm.	✓	✓	✓	✗
Ours (`Prompt Promotion`)	✓	✓	✓	✓

4.3.1 Performance Comparison

The performance of the variants is shown in Figure 4. Note that we skip the *w/ Rand. Perm.* setting for the *Base* variant since the order of two tokens is trivial and randomly permuting them does not affect the performance too much. From Figure 4, we make the following key observations:

- We consider the *Base* variant as training the BERT encoder to replace the matrix multiplication operation in DistMult. While it does not induce model collapse, it is still challenging to enforce a BERT encoder to fill the role of the operation. This observation inspires our `Prompt Promotion` approach, which detours the functionality replication of matrix multiplication and extends the power beyond it by introducing additional prompts.
- The addition of randomly generated prompts completely collapses the model, regardless of the use of random permutation. This is because the model is overwhelmed with not only the HGC task, but also the identification of the queried entity and relation tokens. This suggests the requirements of carefully crafted prompts with very limited noises.



(a). Full Prompt (Ours) (b). Metapath-based Only (c). Embedding-based Only

Figure 5: Learning dynamics of models with full prompts, metapath-based prompts only, and embedding-based prompts only.

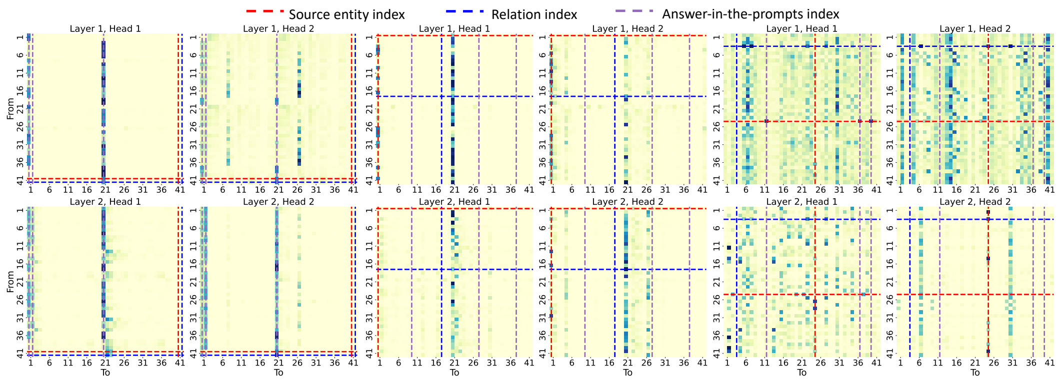
- The *Embedding-based Only* variant yields decent performance under the two settings, especially for the hit ratios with small K's. This not only validates the necessity of the embedding-based prompts, but also confirms one side of the hypothesis - the BERT structure is considerably good at identifying the existing answer among the input prompts.
- The fact that *Metapath-based Only* underperforms the *Base* can also be explained by the unavoidable noise introduced in the prompts. In comparison, although *Embedding-based Only* also takes extra prompts, these prompts are structurally similar in the embedding space, while the noise introduced by merely following the metapaths is intractable.
- *Full-Prompt* outperforms all other variants under the two settings, suggesting the necessity in the combination of the two sets of prompts. We also discover that as K increases, the gap between our variants and the baselines decreases faster under the *w/o Rand. Perm.* setting, compared with the other. This is because the model relies too much on identifying the existing prompts by splitting less explanation power in deducing the query patterns. Randomly permuting the input tokens mingles the prompts all together, therefore forcing the model to focus on the intrinsic connection between the prompts and the query, rather than the one hooked by the token positions.

4.3.2 Training Dynamics Analysis

Table 3: Performance comparison with the baselines. Best results are bolded, and runner-ups are underlined.

Model	Hit@1	Hit@3	Hit@5	Hit@10	MRR
DisMult (Yang et al., 2015)	.6040	.7280	.7550	.8350	.6840
ComplEX (Trouillon et al., 2016)	.6680	.7780	.8180	.8650	.7370
ConvE (Dettmers et al., 2018)	.6400	.7460	.7950	.8490	.7110
HittER (Chen et al., 2021)	.5505	.6758	.7227	.7862	.6312
ConvE-LTE (Zhang et al., 2022)	.6350	.7444	.7918	.8506	.6602
Distmult-LTE (Zhang et al., 2022)	.6381	.7651	.8083	.8677	.7174
Base	.7246	.7610	.7729	.7895	.7481
Emb.-based Only	.7786	.8272	.8447	.8672	.8096
Mtp.-based Only	.4567	.4740	.4843	.5082	.4795
Ours w/o. Rand. Perm.	.7383	.7817	.7940	.8118	.7653
Ours	**.8393**	**.8710**	**.8802**	**.8922**	**.8587**

We analyze the training dynamics of the *Embedding-based Only*, *Metapath-based Only*, and *Full-Prompt* variants under two settings with their learning curves shown in Figure 5. Comparing from the setting perspective, we observe that models converge slower under *w/ Rand.Perm.*. This is because variants under *w/o Rand.Perm.* tends to take the shortcut solution by memorizing the positions, rather than learning the behavior patterns. Identifying the shortcut token's positions, compared with the HGC task, is a relatively easier task that requires less model complexity and learning time. This aligns with our "dual-task" hypothesis - the easier line of task is to identify the answer from the prompts, leading to faster convergence, and the harder one is to deduce the query patterns, corresponding to a relatively slower convergence. Additionally, we find that the gaps in hit ratios for



(a). Train and test w/o Rand. Perm. (b). Train w/o and test w/ Rand. Perm. (c). Train and test w/ Rand. Perm (ours).

Figure 6: Attention heatmaps of the case under three settings.

different K's are larger under *w/ Rand. Perm.*, indicating better generality and pattern extrapolation abilities. Among the variants, *Full-Prompt* exhibits reasonable learning behavior. It avoids saturating too quickly like *Metapath-based Only* due to less introduced noise, and does not take excessively long to achieve performance improvement like *Embedding-based Only*, which relies heavily on accessible shortcuts that hinder generality.

4.4 Random Permutation on Model Learning

To further analyze how the random permutation affects the model learning, we empirically study the model behavior under a specific query case. Consider the *Full-Prompt* variant, where we differ the train and test conditions: (a) We train and test the variant under *w/o. Rand. Perm.*.; (b) We train the variant *w/o. Rand. Perm.*, but test it under *w/ Rand. Perm.*; (c) We train and test the variant under *w/ Rand. Perm.*. Regarding a specific query, we show the normalized attention scores heat map under the three conditions in Figure 6. The rankings of the correct answer under the three conditions are 1, 133, and 1 respectively. Under condition (a), we see the model consistently pay heavy attention to tokens on positions 1 and 21. This is because we set m=20 and n=20, and the most probable answers can usually be found in these positions. Without random permutation, the model quickly identifies the shortcut, rather than paying extra attention to the query (indexed by the red and blue dotted lines). Under condition (b), the model failed to assign a high ranking to the correct answer. Due to the random permutation, tokens on positions 1 and 21 no longer provide precise information as the model assumes, making it overwhelmed with the introduced randomness. This can also be confirmed with small attention scores assigned to the query and the potential answers. Therefore, randomly permuting the input sequence acts as a potential and effective attack to variants trained under *w/o Rand. Perm.*. The model trained and tested under *w/ Rand. Perm.* as we designed, on the other hand, assigns much more even attention to the input sequence. More specifically, it learns to assign attention to the potential answers in the input (red and purple line intersections in Layer 1, Head 1), the source entity (red vertical dotted line in Layer 2, Head 2), as well as other important information in deems important (tokens indexed by 7, 31, etc.). This confirms that random permutation enhances the model's ability to learn the intrinsic connection between the query and the answer, reducing reliance on input prompts and increasing robustness and generality.

5 Conclusion

In this work, we focus on the heterogeneous graph completion task in the context of app promotion, and propose a prompt-based approach named `Prompt Promotion` that leverages a pre-trained BERT to model the connection patterns in the complex app promotion ecosystem. Specifically, by incorporating both embedding-based and metapath-based prompts, our model first unlocks the prompt learning for app promotion graphs, and achieves superior performance compared to baselines. In addition, we conduct thorough analysis regarding the components, training dynamics to illustrate the delicacy of our designed framework. The contributions of this research include advancing the understanding of app promotion networks, improving trustworthiness in recommender systems, and detecting promotion traces of malicious apps. Future directions involve exploring additional prompt generation strategies and further enhancing the model's performance.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2, 6, 14

Sanxing Chen, Xiaodong Liu, Jianfeng Gao, Jian Jiao, Ruofei Zhang, and Yangfeng Ji. 2021. HittER: Hierarchical Transformers for Knowledge Graph Embeddings. In *Conference on Empirical Methods in Natural Language Processing*. 6, 8, 13, 14

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2D Knowledge Graph Embeddings. In *AAAI Conference on Artificial Intelligence*. 6, 8, 13, 14

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*. Association for Computational Linguistics. 4

Michael C Grace, Wu Zhou, Xuxian Jiang, and Ahmad-Reza Sadeghi. 2012. Unsafe exposure analysis of mobile in-app advertisements. In *ACM conference on Security and Privacy in Wireless and Mobile Networks*. 1

Michaela Hardt and Suman Nath. 2012. Privacy-aware personalization for mobile advertising. In *ACM conference on Computer and communications security*. 1

Ling Jin, Boyuan He, Guangyao Weng, Haitao Xu, Yan Chen, and Guanyu Guo. 2021. MAdLens: Investigating into Android In-App Ad Practice at API Granularity. *IEEE Transactions on Mobile Computing* (2021). 1

Bin Liu, Bin Liu, Hongxia Jin, and Ramesh Govindan. 2015. Efficient privilege de-escalation for ad libraries in mobile apps. In *Annual International Conference on Mobile systems, Applications, and Services*. 1

Tianming Liu, Haoyu Wang, Li Li, Xiapu Luo, Feng Dong, Yao Guo, Liu Wang, Tegawendé Bissyandé, and Jacques Klein. 2020. MadDroid: Characterizing and detecting devious ad contents for android apps. In *The Web Conference*. 1

Xin Lv, Yankai Lin, Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Peng Li, and Jie Zhou. 2022. Do pre-trained models benefit knowledge graph completion? a reliable evaluation and a reasonable approach. In *Findings of the Association for Computational Linguistics*. 2, 14

Suman Nath. 2015. Madscope: Characterizing mobile in-app targeted ads. In *Annual International Conference on Mobile Systems, Applications, and Services*. 1

Omid Rafieian and Hema Yoganarasimhan. 2021. Targeting and privacy in mobile advertising. *Marketing Science* (2021). 1

Google Research. 2023. How people discover, use, and stay engaged with apps. *Think with Google* (2023). 1

Sooel Son, Daehyeok Kim, and Vitaly Shmatikov. 2017. What Mobile Ads Know About Mobile Users. Internet Society. 1

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2018. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *International Conference on Learning Representations*. 2, 14

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *International Conference on Machine Learning*. 6, 8, 13, 14

Narseo Vallina-Rodriguez, Jay Shah, Alessandro Finamore, Yan Grunenberger, Konstantina Papagiannaki, Hamed Haddadi, and Jon Crowcroft. 2012. Breaking for commercials: characterizing mobile advertising. In *Internet Measurement Conference*. 1

Nicolas Viennot, Edward Garcia, and Jason Nieh. 2014. A measurement study of google play. In *ACM international conference on Measurement and modeling of computer systems*. 1

Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. 2021. Structure-augmented text representation learning for efficient knowledge graph completion. In *The Web Conference*. 2, 14

Xin Xie, Ningyu Zhang, Zhoubo Li, Shumin Deng, Hui Chen, Feiyu Xiong, Mosha Chen, and Huajun Chen. 2022. From discrimination to generation: knowledge graph completion with generative transformer. In *The Web Conference*. 14

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *3rd International Conference on Learning Representations, ICLR*. 2, 4, 6, 7, 8, 13, 14

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for knowledge graph completion. *arXiv preprint arXiv:1909.03193* (2019). 2, 14

Zhanqiu Zhang, Jie Wang, Jieping Ye, and Feng Wu. 2022. Rethinking Graph Convolutional Networks in Knowledge Graph Completion. In *The Web Conference*. 6, 8, 13

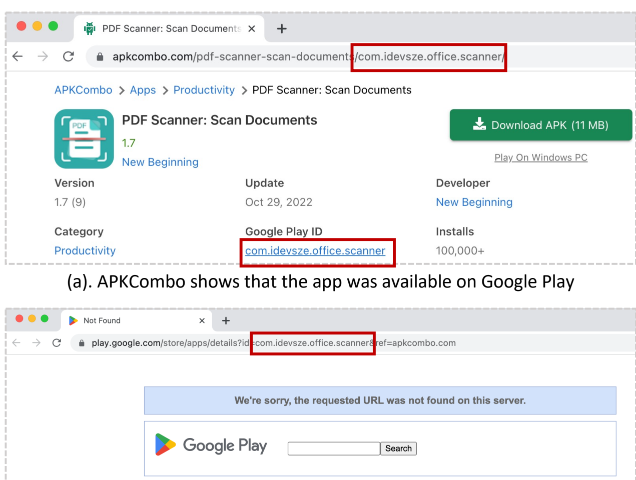
A App Promotion Heterogeneous Graph

Table 4 shows the statistics of the entities in the constructed App Promotion Heterogeneous Graph (APHG). In addition, we define the relations as follows: (1) R1: an `app-promote-app` relation indicates that there exists a promotion link from the subject app to the object app; (2) R2: an `app-include-signature` relation means that a digital signature can be used to verify the authenticity and integrity of the app package; (3) R3: an `engine-detect-app` relation indicates that a VT engine marks an app with a specific flag (e.g., adware or Trojan); (4) R4: an `app-belong-category` represents that an app belongs to a specific app category categorized by Google Play; (5) R5: a `developer-involve-category` relation suggests that an app created by the developer is categorized into a specific app category; (6) R6: a `developer-develop-app` relation signifies that a developer develops an app; (7) R7: an `app-access-URL` relation denotes that an app has access to a specific URL; (8) R8: a `developer-use-URL` relation indicates that the app developed by the developer may access a specific URL; (9) R9: an `app-own-manifest` relation represents that an app is associated with a specific manifest file. Since apps with different security levels follow different behavior patterns, we further divide the apps into three classes. For example, the relation `app-belong-category` is extended to three relations: `benign-belong-category`, `grey-belong-category`, and `mal-belong-category`. As a result, the above relations are extended to twenty-nine classes of relation types.

Table 4: Numbers and types for entities (or nodes).

type	**Signature**	**VT Engine**	**Category**	**Developer**	**URL**
num.	185	65	36	3139	18870
type	**Manifest**	**Benign**	**Greyware**	**Malware**	**Total**
num.	10269	3961	1143	363	38031

B Illustration of Information Scarcity



(a). APKCombo shows that the app was available on Google Play

(b). Google Play server cannot find the app anymore

Figure 7: Illustration of information scarcity on Google Play.

We here provide an illustration of information scarcity of within the app promotion ecosystem. As illustrated in Figure 7, the app *PDF Scanner*, which acts as a seed app and plays an instrumental role in promoting subsequent apps, was once available on Google Play. However, by relying solely on Google Play as our information source, we inevitably encounter instances where certain attributes, such as those related to the developer, are absent. Such omissions of information substantially impede our capacity to decipher patterns in application promotion behavior, and further motivate us to target the HGC task on our app promotion graph.

Algorithm 1 `Prompt Promotion`: a simplified PyTorch-style Pseudocode of our method on the HGC task.

```
# model: BERT-based model
# pretrained_kge: pretrained KGE method
# M: filtered metapaths for each relation
# Train model for N epochs
for query, target in dataloader:
  # Obtain emb-based prompts
  emb_prompt = pretrained_kge(query)[:n]

  # Find all reachable entities
  reached_ent = follow_metapath(query, M)

  # Sample k metapath-based prompts
  mtp_prompt = sample(reached_ent, m)

  # Forward
  input_seq = rand_perm(concat(emb_prompt, mtp_prompt, query))
  pred = model(input_seq)
  loss = CrossEntropyLoss(pred, target)

  # Optimize model with loss backward
  loss.backward()
  optimizer.step()
```

C Pseudo Code for Prompt Promotion

We provide the PyTorch style pseudocode of our proposed `Prompt Promotion` in Alg. 1 over the app promotion HGC task.

D Experimental Setups

D.0.1 Dataset

Our app promotion dataset is collected from AndroZoo, a well-maintained and regularly updated repository that provides various versions of apps from official app markets like Google Play. The dataset encompasses apps released between January 1st, 2018, and February 3rd, 2023. We classify the apps into three categories based on the number of engines that flag them on VirusTotal. Malware apps are flagged by at least 10 engines, greyware apps are flagged by 1 to 9 engines, and benign apps are not flagged by any engine on VirusTotal. Our seed dataset comprises approximately 48,000 apps, evenly distributed among the three classes, providing a diverse set of apps representing different levels of potential security risks. More details regarding the dataset and the construction for APHG are provided in Section 2.3.

D.0.2 Baselines

We compare our approach against several baseline models commonly used in the graph completion task:

- **DistMult** Yang et al. (2015): DistMult represents entities and relations as low-dimensional vectors and utilizes a bilinear dot product scoring function for link prediction.
- **ComplEX** Trouillon et al. (2016): ComplEX extends DistMult by using complex-valued embeddings, allowing for a more expressive representation and remaining linear in both space and time.
- **ConvE** Dettmers et al. (2018): ConvE employs a convolutional neural network architecture to encode entities and relations. It operates on 2D tensors to capture local patterns and dependencies within the knowledge graph.
- **HittER** Chen et al. (2021): HittER utilizes hierarchical transformers to learn knowledge graph embeddings, balancing the contextual relational information and the information from the training entity.
- **LTE** Zhang et al. (2022): LTE extends embedding-based methods by equipping existing knowledge graph embedding models with linearly transformed entity embeddings. It mines semantic

information from entity representations to enhance the model performance. In this paper, we select DistMult and ConvE as the backbones, denoted as DistMult-LTE and ConvE-LTE respectively.

D.0.3 Evaluation Metrics

We evaluate the graph completion performance using two key metrics: mean reciprocal rank (MRR) and Hits@K. We empirically set the beam size for MRR as 256.

D.0.4 Implementation Details

We use a pre-trained DistMult Yang et al. (2015) as the backbone model to tokenize the entities and relations as low-dimensional vectors. Note that this choice is a matter of preference, and can be substituted with other embedding-based methods such as TransE Bordes et al. (2013). ComplEX Trouillon et al. (2016) is utilized for prompt filtering, and can also be replaced by any other graph completion methods. We encode the input sequence with a two-layer BERT model, and utilize the sum operation to aggregate the encoded sequence. Finally, a two-layer MLP is applied as the prediction head for the HGC task. During training, we employ the AdamW optimizer and use binary cross-entropy as the loss function. The learnable parameters of the pre-trained DistMult are initialized randomly, while BERT is loaded with pretrained weight parameters. The training process is conducted on an NVIDIA RTX 3090 GPU with 24 GB of memory.

E Related Work

For the task of graph completion/link prediction, methods that learn both the entity and relation representations are categorized into embedding-based and transformer-based, depending on their intrinsic modeling structures.

Embedding-based Methods. Knowledge graph embedding (KGE) methods employ geometric operations in the vector space to capture the underlying semantics of the graph, such as translation Bordes et al. (2013), bilinear transformation Yang et al. (2015), rotation Sun et al. (2018). Other methods design embeddings from different perspectives. For instance, CompLEX Trouillon et al. (2016) leverages compositionality to model the complex relationships between entities. ConvE Dettmers et al. (2018) utilizes multi-layer convolutional networks on the 2D grid abstracted from the knowledge graph to encode local dependencies. Although conceptually straightforward, these methods encode each entity and relation's embedded information through a simple vector. The inherent simplicity of embedding-based methods can present challenges in scenarios involving complex reasoning and scarcity of information.

Transformer-based Methods. Taking account of the relatively weak expression power of the embedding-based methods, several recent works utilize transformers for additional enhanced contextual information encoding. Some works take the triple as the input and perform tasks such as triple classification and link prediction. For example, KG-BERT Yao et al. (2019) treats triples as textual sequences to inject semantic information and exploits pretrained BERT to learn context-aware embeddings. PKGC Lv et al. (2022) leverages the entity's semantic information and converts them into natural prompt sentences to address the closed-world assumption (CWA) and incoherent issue. However, the above methods require the scoring of all possible triples in inference, therefore introducing some unnecessary calculation overheads. On the other head, some other works are designed to directly output the candidate entities. For example, StAR Wang et al. (2021) designs a structure-aware and structure-augmented framework for efficient KGC inference. HittER Chen et al. (2021) extracts context neighbors for the source entity and introduces the additional masked entity prediction task for balanced contextualization. GenKGC Xie et al. (2022) introduces relation-aware demonstration and entity-ware hierarchical decoding for better representation learning. Despite the progress made so far, we notice some implementation gaps in applying the above methods to a knowledge graph and a heterogenous graph: First, entities in a knowledge graph naturally entitle semantic information, while this is not always true for a heterogeneous graph; Second, the above methods left out the entity/node type information provided in a heterogeneous graph, therefore leaving considerate space for performance improvement. In contrast, our model is designed to not only straightly output the candidate entities, which eliminates the calculation overhead, but also fully utilize the entity and relation type information for better prompting.