Reinforced Anchor Knowledge Graph Generation for News Recommendation Reasoning

Danyang Liu
University of Science and Technology of China
Hefei, China
ldy591@mail.ustc.edu.cn

Jianxun Lian*
Microsoft Research Asia
Beijing, China
jianxun.lian@microsoft.com

Zheng Liu
Microsoft Research Asia
Beijing, China
zheng.liu@microsoft.com

Xiting Wang
Microsoft Research Asia
Beijing, China
xitwan@microsoft.com

Guangzhong Sun
University of Science and Technology of China
Hefei, China

Xing Xie
Microsoft Research Asia
Beijing, China
xing.xie@microsoft.com

ABSTRACT

News recommendation systems play a key role in online news reading service. Knowledge graphs (KG), which contain comprehensive structural knowledge, are well known for their potential to enhance both accuracy and explainability. While existing works intensively study using KG to improve news recommendation accuracy, using KG for news recommendation reasoning has not been fully explored. A few works such as KPRN [18], PGPR [22] and ADAC [25] have discussed knowledge reasoning in some other recommendation domains such as music or movie, but their methods are not practical for the news. How to make reasoning scalable to generic KGs, easy to deploy for real-time serving and meanwhile elastic for both recall and ranking stages remains an open question.

In this paper, we fill the research gap by proposing a novel recommendation reasoning paradigm AnchorKG. For each article, AnchorKG generates a compact Anchor Knowledge Graph, which corresponds to a subset of entities and their k-hop neighbors in the KG, restoring the most important knowledge information of the article. On one hand, the anchor graph can be used to enhance the latent representation of the article. On the other hand, the interaction between two anchor graphs can be used for reasoning. We develop a reinforcement learning-based framework to train the anchor graph generator, in which there are three major components, including the joint learning of recommendation and reasoning, sophisticated reward signals, and a warm-up learning stage. We conduct experiments on one public dataset and one private dataset. Results demonstrate that the AnchorKG framework not only improves recommendation accuracy, but also provides high quality knowledge-aware reasoning. We release the source code at https://github.com/danyang-liu/AnchorKG .

*Corresponding author.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467315

CCS CONCEPTS

• **Information systems** → **Document representation**; **Recommender systems**; **Data mining**; • **Computing methodologies** → **Knowledge representation and reasoning**.

KEYWORDS

news recommender, knowledge graph, recommendation reasoning

ACM Reference Format:
Danyang Liu, Jianxun Lian, Zheng Liu, Xiting Wang, Guangzhong Sun, and Xing Xie. 2021. Reinforced Anchor Knowledge Graph Generation for News Recommendation Reasoning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467315

1 INTRODUCTION

Online news reading services provide fresh information feeds for people and become an indispensable part of people's daily lives. To build a more intelligent news recommender system, researchers have long been eager to leverage knowledge graphs (KG) for boosting models' accuracy [6, 13, 14]. However, besides the capability of improving recommendation accuracy, knowledge graphs also possess the ability to enhance recommendation explainability, which, in fact, is more critical and challenging, but less explored.

In this paper, we focus on a crystal type of explainability known as *reasoning*, in which explainable reasons are in the pattern of multi-hop relational paths over a KG. In this line of research, [18] proposes a knowledge-aware path recurrent network (KPRN) for improving recommendation accuracy and providing reasoning by constructing qualified paths between the user and the item. However, KPRN can only be applied to small knowledge graphs with very limited types of relations. In the news domain, the associated KG contains thousands of relations, it is infeasible to fully enumerate all the paths for each user-item pair. To avoid enumerating all paths, PGPR [22] and ADAC [25] introduce reinforcement learning-based methods to perform explicit reasoning. However, both PGPR and ADAC suffer from several weaknesses. First, they seek a single path between the source user and the target item. Usually, a news article contains multiple entities. A single path may not be able to fully reveal the relationship between the source and target objects. Second, the

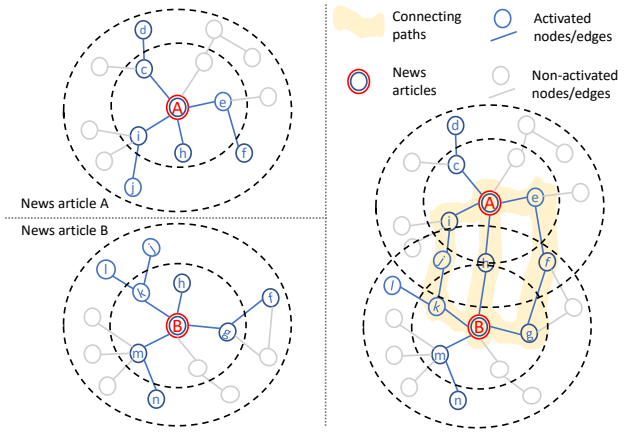


Figure 1: A toy example for illustrating the AnchorKG. Each news article will generate one anchor knowledge graph independently. When discovering their relationships, the anchor graphs will be bridged by the overlapped entities and reasoning paths (in yellow) will be constructed accordingly.

textual content of news articles is not well exploited. In addition, they can only be applied to the recall stage of recommender systems but are not able to score or reason a designated user-item pair (a typical ranking stage), which limits their flexibility.

To address these issues, we propose a novel recommendation reasoning paradigm named Anchor Knowledge Graph (**AnchorKG**). The basic idea is that, for each news article, we leverage a policy network to generate a subgraph from the KG. This subgraph is supposed to contain key entities in the news article, as well as some necessary neighboring entities which connect the article entities with multi-hop relational paths. We call this knowledge subgraph an *anchor graph* because it only selects the most important knowledge entities from the exponential growth of multi-hop relations and serves as a footprint for the news article over the KG. Once anchor graphs are generated, the knowledge-aware reasoning of any two news articles can be conducted simply based on the interactions of their corresponding anchor graphs. Figure 1 illustrates the concept of AnchorKG. The advantages of the proposed paradigm are three-fold. First (accuracy), the knowledge information conveyed in anchor graphs are concentrated (only important entities are selected from all the connected paths) and deep (it contains multi-hop useful relational paths), so anchor graphs can enrich the representation of news articles and improve the accuracy of news recommendations. Second (flexibility), the framework is applicable for different stages of recommender systems, including the recall stage and ranking stage. Meanwhile, reasoning results between two items are not limited to a single path pattern anymore. Third (scalability), the construction of an anchor graph only depends on the news article itself, so the construction process can be fully pre-computed and cached in the offline stage, then used for online serving, which leaves a possibility for large-scale real time services.

How to train the anchor graph generator is non-trivial. Motivated by [22, 25], we cast the anchor graph generation as a deterministic Markov Decision Process (MDP) and adopt reinforcement learning techniques to optimize the framework. We regard each news article as a virtual node in a KG, which connects to normal knowledge entities with a special edge "*mention*". The agent starts from a virtual node and expands over the KG, with a policy network to determine whether the navigated node should be absorbed or ignored until the anchor graph reaches a certain size or a maximum level of hops. There are three main challenges in the model training: (C1) how to guarantee that the selected nodes can provide high-quality explainability; (C2) how to make the model converge more quickly to a satisfying solution; (C3) how to coordinate the reasoning paths and item recommendations, so that the discovered reasoning paths are not just a post-hoc, uncoupled explanation for the recommended items. To solve C1, we design immediate and final rewards with knowledge-aware negative sampling for the MDP states. These rewards can guide the policy network to expand towards meaningful nodes rather than towards random or trivial nodes. Meanwhile, we design a mechanism to select hard negative pairs from KG, to further encourage the model to select more distinguishable nodes. To solve C2, we adopt the behavior cloning [8] technique to warm start the model training. Specifically, we construct a small dataset and train a KPRN [18] model to generate reasoning paths between source-target pairs. The warm start stage is an imitation learning process, where KPRN's reasoning paths will serve as ground-truth labels. As for C3, we develop a multi-task alternative learning process to optimize both the recommendation task and the anchor graph generation task jointly. We conduct comprehensive experiments on two news recommendation datasets. Results demonstrate that the AnchorKG can not only improve the model accuracy but also provide high-quality reasoning paths over the KG. To summarize, we make the following major contributions:

- We propose a brand-new paradigm, AnchorKG, for news recommendation reasoning over a knowledge graph. At its core is an anchor graph generator, which generates a small topic-aware subgraph from a general knowledge graph to cover key entities and relations for the given news article. The anchor graph can be used for both enhancing document representation and providing knowledge reasoning between news documents.
- We develop a reinforcement learning-based optimization framework to train the AnchorKG. The training framework includes three major parts: a joint learning of recommendation task and graph generation task, sophisticated reinforcement reward signals, and a warm start stage with behavior cloning.
- Experiments on two real-world news datasets demonstrate that AnchorKG not only achieves best accuracy performance but also provides high-quality reasoning knowledge paths.

2 METHODOLOGY

We first introduce some terminologies before describing our proposed method.

Item-to-Item Recommendation. In this paper, we consider the practical scenario of the item-to-item (I2I) recommendations, where we aim to discover related items $\{n_j\}$ for a given item n_i. I2I plays an important role in recommendation systems, for instance, when users finish reading one news articles, at the bottom of the page, I2I provides some related articles for continue reading; in an industrial two-stage (recall and ranking) system, I2I is frequently used as a candidate recall component.

Knowledge Graph. The knowledge graph is organized as a set of semantic triples: $\mathcal{G} = \{(e_h, r, e_t)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$ and $\mathcal{R}$ represent the set of entities and relations, respectively. We resort to TransE [1] to learn a fixed low-dimensional representation vector for entities and relations.

k-Hop Path A k-hop path is a continuous trajectory connecting a source entity e_0 and target entity e_k over the knowledge graph by k relations: $p^k = \{e_0, r_1, e_1, r_2, e_2, \cdots, r_k, e_k|e_* \in \mathcal{E}, r_* \in \mathcal{R}\}$.

Knowledge-Aware Recommendation Reasoning. With the help of knowledge graph $\mathcal{G}$, given an item n_a, we recommend a list of items $\Omega_a = \{n_1, n_2, ..., n_k\}$ related to it; meanwhile, for each pair $\langle n_a, n_i \rangle$, where $n_i \in \Omega_a$, we provide a list of multi-hop paths $\Delta_{a \Leftrightarrow i} = \{p_1, p_2, ..., p_m\}$ connecting n_a and n_i over $\mathcal{G}$, which can be used to reason the potential relationships between the item pair.

2.1 A New Paradigm for Reasoning

We propose a new reasoning framework AnchorKG, which is short for Anchor Knowledge Graph generation for recommendation reasoning, with the goal of achieving the following properties:

(1) The framework can seamlessly incorporate textual content and knowledge graphs for both recommendation and reasoning.
(2) The reasoning formulation is not limited to a single path but can include multiple paths.
(3) The framework is flexible to be applied to different components of recommendation systems, such as the recall stage and the ranking stage.
(4) The framework is applicable to serve real-time news recommendations with large-scale knowledge graphs.

Anchor Graph. At the core of AnchorKG is the generation of anchor graphs. The anchor graph $\mathcal{X}_a = (\mathcal{E}_a, \mathcal{R}_a)$ of a given news article n_a is a subgraph extracted from the knowledge graph $\mathcal{G}$, where $\mathcal{E}_a \subseteq \mathcal{E}$ and $\mathcal{R}_a \subseteq \mathcal{R}$. $\mathcal{X}_a$ containing part of or all the entities mentioned in n_a as well as a selected set of the entities' most important k-hop paths, $k \in [1, K]$ where K is the maximum allowed number of hops. $\mathcal{X}_a$ can be regarded as a footprint of n_a over the knowledge graph. $\mathcal{X}_a$ provides valuable auxiliary information to the news content. Therefore, it can be incorporated to enhance the document representation with a knowledge fusion module $\mathcal{H}_\psi$: $\widehat{v}_a = \mathcal{H}_\psi(v_a, \mathcal{X}_a)$. The updated document vector $\widehat{v}_a$ extends the capability of measuring the similarity between two news articles, which eventually leads to better recommendation accuracy.

Reasoning. Once the anchor graph for each news article has been generated, to reason the relationship between two articles n_a and n_b, we can simply count the overlapped entities of the corresponding anchor graphs $\mathcal{X}_a$ and $\mathcal{X}_b$. Through the overlapped entities we can establish high-order connectivity between n_a and n_b, in the form of a collection of multi-hop paths: $\Delta_{a \Leftrightarrow b} = \{p_0, p_1, ...\}$. Each path in $\Delta_{a \Leftrightarrow b}$ can be regarded as an evidence for explainable reasoning.

2.2 Anchor Graph Generation

We need a subgraph generation model π_θ to construct an anchor graph $\mathcal{X}_a$ for each news article n_a. An intuitive method is to adopt a model like KPRN [18], where we first enumerate all the possible k-hop paths starting from entities of n_a, then train a neural attention model with supervised learning methods to select a few most important paths. However, in the news recommendation domain, the associated knowledge graph usually contains thousands of relations and millions of entities. A brute-force path enumeration process like KPRN is very complex and unpractical. Moreover, we hope the anchor graph generation of different articles can be individually independent, without any contextual requirements (e.g., in KPRN, we need to provide a target article before searching the qualified paths). To this end, we resort to reinforcement learning techniques to learn an anchor graph generator π_θ. The basic idea is, we train an agent that starts from n_a (which can be regarded as a virtual node in the knowledge graph, connecting to all entities mentioned in the article), then iteratively expands the subgraph by examining whether the connecting neighbor nodes should be absorbed into the subgraph, until the subgraph reaches a stopping criterion. An overview of the proposed framework is illustrated in Figure 5 in Appendix, with components formulated as follows.

2.2.1 The Markov Decision Process. We formulate the anchor graph generation problem as a Markov Decision Process (MDP) [11]: $M = \{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}\}$, where $\mathcal{S}$ is a finite set of states during exploration, $\mathcal{A}$ is a finite set of actions, $\mathcal{R}$ is the reward function from the environment, and $\mathcal{P}$ is the transition probability function.

State: The state $s_t \in \mathcal{S}$ at step t is defined as a tuple $(n_a, \mathcal{E}_t, \mathcal{R}_t, E_t)$, where n_a is the source news article, $\mathcal{E}_t$ and $\mathcal{R}_t$ denote the set of all the absorbed entities and relations up to step t, E_t denotes the newly absorbed entities at step t. The initial state $s_0 = (n_a, \{e_a\}, \emptyset, \{e_a\})$, where $\emptyset$ indicates an empty set and e_a is the virtual node of the article itself. e_a connects to all entities which appear in the article n_a with a special type of edge "*mentioned*".

Action: For state s_t at step t, its action space $\mathcal{A}_{s_t}$ is the set of outgoing relationships of entities in E_t: $\mathcal{A}_{s_t} = \{(r, e)|(e_h, r, e) \in \mathcal{G}, e_h \in E_t, e \notin \mathcal{E}_t\}$. Starting from the initial state $s_0 = (n_a, \{e_a\}, \emptyset, \{e_a\})$, the agent will leverage a policy network π_θ to predict the most promising outgoing relations and absorb to them the explored set $\mathcal{E}_t$ and $\mathcal{R}_t$, until the size of $\mathcal{E}_t$ reaches a preset condition.

Reward: The reward function $\mathcal{R}$ measures the quality of the navigated states. Different from [22, 25], we don't have a well-defined terminal state (which is the target entity in [22, 25]) serving as ground-truth to provide direct terminal rewards. We define two types of rewards to provide feedback from the environment:

- Immediate reward: at each step t, after the agent performs an action a_t, we will get an immediate reward $\mathcal{R}_I$ based on the newly added entities E_t: $\mathcal{R}_I = g_{RI}(E_t)$.
- Terminal reward: after the agent reaches the final state s_T, we will get a soft task-aware reward on the generated anchor graph by using it to conduct the item recommendation task and the reasoning task: $\mathcal{R}_T = g_{RT}(\mathcal{E}_T, \mathcal{R}_T)$.

The detailed implementation of g_{RI} and g_{RT} will be introduced in Section 2.2.3.

Transition: Given a state $s_t = (n_a, \mathcal{E}_t, \mathcal{R}_t, E_t)$ and an action $a_t = \{(r, e)\} \subseteq \mathcal{A}_{s_t}$, transition to the next state s_{t+1} is deterministic:

$$\mathcal{P}\Big(s_{t+1} = (n_a, \mathcal{E}_{t+1}, \mathcal{R}_{t+1}, E_{t+1})|s_t, a_t\Big) = 1 \tag{1}$$

where $E_{t+1} = \{e|e \in a_t\}$, $\mathcal{E}_{t+1} = \mathcal{E}_t \cup E_{t+1}$, $\mathcal{R}_{t+1} = \mathcal{R}_t \cup \{r|r \in a_t\}$.

2.2.2 AnchorKG Policy Network. We develop a neural policy network that learns the probability distribution of taking actions a given a state s_t: $\pi_\theta(s_t, a_t) = P(a_t|s_t, \theta)$, where θ is the parameter of

the policy network. We project the states and actions into a latent semantic space. Specifically, state s_t is modeled as

$$\mathbf{s}_t = \boldsymbol{v} \oplus \boldsymbol{\mu}_t \oplus \boldsymbol{\omega}_t \tag{2}$$

where $\oplus$ is the vector concatenation operation, $\boldsymbol{v}$ is the document embedding of the current article, $\boldsymbol{\mu}_t$ is the vector representation of the anchor graph at step t, and $\boldsymbol{\omega}_t$ is the vector representation of the newly added entities at step t. To compute $\boldsymbol{\mu}_t$, we first refine the embedding of each entity e_h in $\mathcal{E}_t$ with its neighborhood, then aggregate all the entities with a simple attention network:

$$\boldsymbol{e}'_h = Tanh\left(\boldsymbol{W}_0\left(\boldsymbol{e}_h \oplus \sum_{\substack{(e_h,r,e_t)\in\mathcal{X}_t,\\ or\ (e_t,r,e_h)\in\mathcal{X}_t}} (\boldsymbol{r}+\boldsymbol{e}_t)\right)\right) \tag{3}$$

$$\alpha_0(e_h) = \boldsymbol{w}_2 ELU(W_1 \boldsymbol{e}'_h),\quad \alpha(e_h) = \frac{exp(\alpha_0(e_h))}{\sum_{b\in\mathcal{E}_t} exp(\alpha_0(e_b))} \tag{4}$$

$$\boldsymbol{\mu}_t = \sum_{h\in\mathcal{E}_t} \alpha(e_h)\boldsymbol{e}'_h \tag{5}$$

Where ELU is the exponential linear unit. Since $\boldsymbol{\mu}_t$ provides most of the information from the state, for $\boldsymbol{\omega}_t$ we can use a much simpler method, which is just the average of entity embeddings in E_t:

$$\boldsymbol{\omega}_t = \frac{\sum_{e_h\in E_t} \boldsymbol{e}_h}{|E_t|} \tag{6}$$

Each available action unit $a_i = (r_i, e_i)$ is modeled as the superposition of two vectors:

$$\boldsymbol{a}_i = \boldsymbol{r}_i + \boldsymbol{e}_i \tag{7}$$

Then, the policy network calculates the probability of taking an action unit a_i at state s_t with the following equation:

$$\pi_\theta(s_t, a_i) = \frac{W_3 ReLU(W_4(\boldsymbol{s}_t \oplus \boldsymbol{a}_i))}{\sum_{a_k\in\mathcal{A}_k} W_3 ReLU(W_4(\boldsymbol{s}_t \oplus \boldsymbol{a}_k))} \tag{8}$$

In practice, to accelerate the generation process, we let the agent absorb multiple outgoing relations at one action a_t, instead of only absorbing one relation unit (r, e). Specifically, for every $e \in E_t$, we set a hyper-parameter d, which we call *receptive field size*, to denote the number of outgoing relations to be absorbed. A small trick is that we add a self-loop to connect an entity to itself. If the self-loop is selected (by once or multiple times), the absorbed outgoing relations for this entity will be fewer than d. At different levels of k-hop paths, the receptive field sizes can be different. In practice, the best setting is $[5, 3, 2]$, which means that receptive field sizes for 1-hop, 2-hop and 3-hop nodes are 5, 3 and 2, respectively. At the first layer (in which entities are directly connected to the given source node), we select at most 5 neighbors; at the second layer (in which the entities are 2-hop away from the given source node), we will absorb at most 5×3 neighbors in total, and so on.

2.2.3 Implementation of Rewards. In Section 2.2.1, the rewards are comprised of two components: the terminal reward and the immediate reward. Here we give a detailed definition for them.

Terminal Reward: After the generation process finishes, we need to evaluate the quality of the generated anchor graph. Because we expect to use the anchor graph to help both recommendations and reasoning, we rely on the feedback from these two tasks to define the terminal reward. Specifically, we have:

- Recommendation Reward. The anchor graph will help to enhance the document representation: $\widehat{\boldsymbol{v}}_a = \mathcal{H}_\psi(\boldsymbol{v}_a, \mathcal{X}_a)$. We encourage the anchor graph to adjust the document representation, so that related item pairs become closer in the latent space, and unrelated item pairs become less close. The reward can be formulated as

$$\mathcal{R}_T^{(1)} = \mathbb{E}_{i\sim\kappa(a)} Sim(\widehat{\boldsymbol{v}}_a, \widehat{\boldsymbol{v}}_i) - \beta \cdot \mathbb{E}_{j\sim\neg\kappa(a)} Sim(\widehat{\boldsymbol{v}}_a, \widehat{\boldsymbol{v}}_j) \tag{9}$$

  where $\kappa(a)$ indicates the set of related documents (positive labels) to the article n_a and $\neg\kappa(a)$ indicates the documents not in $\kappa(a)$ (negative labels). $Sim(\cdot,\cdot)$ is a similarity measurement, we use cosine similarity in this paper. β is a weighting coefficient to adjust the impact of negative samples, which is set to 0.2 empirically in the experiments.
- Reasoning Reward. The reasoning process is established based on the interactions between two anchor graphs. Suppose the anchor graph for article n_a is ready as $\mathcal{X}_a = (\mathcal{E}_a, \mathcal{R}_a)$. The reasoning reward is defined as:

$$\begin{aligned}\mathcal{R}_T^{(2)} =& \mathbb{E}_{i\sim\kappa(a)}\left[\tanh\left(\frac{|\mathcal{E}_a\cap\mathcal{E}_i|}{log(|\mathcal{E}_a|\cdot|\mathcal{E}_i|)}\right) + \tanh\left(\frac{score_\psi(\mathcal{X}_a,\mathcal{X}_i)}{log(|\mathcal{E}_a|\cdot|\mathcal{E}_i|)}\right)\right] \\ &- \beta\cdot\mathbb{E}_{j\sim\neg\kappa(a)}\left[\tanh\left(\frac{|\mathcal{E}_a\cap\mathcal{E}_j|}{log(|\mathcal{E}_a|\cdot|\mathcal{E}_j|)}\right) + \tanh\left(\frac{score_\psi(\mathcal{X}_a,\mathcal{X}_j)}{log(|\mathcal{E}_a|\cdot|\mathcal{E}_j|)}\right)\right]\end{aligned} \tag{10}$$

  where tanh is the hyperbolic tangent function, $|\cdot|$ indicates the cardinality of a set, $\cap$ is the intersection operation, $score_\psi$ measure the quality of overlapped part of $\mathcal{X}_a$ and $\mathcal{X}_i$, which will be instantiated in Section 2.3. Eq.10 encourages related items to have more high-quality overlapping elements in their anchor graphs, while let unrelated items have as few overlap as possible in their anchor graphs. Similar to Eq.9, β is set to 0.2 .

Combining the recommendation reward and reasoning reward, we can get the terminal reward:

$$\mathcal{R}_T = \alpha_1 \cdot \mathcal{R}_T^{(1)} + (1-\alpha_1)\cdot\mathcal{R}_T^{(2)} \tag{11}$$

where $\alpha_1 \in [0, 1]$ is a hyper-parameter.

Immediate Reward: Terminal Reward can only be acquired until the agent reaches an ending state. To make the policy network converge better, we design some immediate rewards. For the sake of notation simplicity, suppose the selected action at step t absorbs only one relation (r, e), which means $a_t = \{(r, e)\}$. The immediate reward is comprised of two parts:

- Coherence Reward. We encourage the agent to select entities which have similar semantic topics (we call it coherence) to the current news article n_a. To this end, we build a reverse index to track all the articles that mention an entity e in the news title: $\Gamma(e) = \{n_1, n_2, \cdots, n_k | n_i\ contains\ e\}$. The coherence reward:

$$\mathcal{R}_I^{(1)}(n_a, e) = cosine\left(\boldsymbol{v}_a, \frac{\sum_{n_i\in\Gamma(e)}\boldsymbol{v}_i}{|\Gamma(e)|}\right) \tag{12}$$

- Hit Reward. We hope the entities mentioned by n_a's related items can be absorbed into the anchor graph. Thus, the hit reward is defined as whether there is at least one document in $\kappa(a)$ which mentions entity e :

$$\mathcal{R}_I^{(2)}(n_a, e) = \begin{cases} 1 & if\ |\{n_i | n_i\in\kappa(a)\wedge n_i\ contains\ e\}| \geq 1 \\ 0 & otherwise \end{cases} \tag{13}$$

The immediate reward can be easily extended to actions a_m which contain multiple relations by accumulating the immediate reward of every single relation:

$$\mathcal{R}_I = \sum_{e_i \in a_m} \left(\mathcal{R}_I^{(1)}(n_a, e_i) + \mathcal{R}_I^{(2)}(n_a, e_i) \right) \tag{14}$$

2.3 Item-to-item Recommendation

Once the anchor graph $\mathcal{X}_a$ of article n_a is generated, we can use $\mathcal{X}_a$ to enhance the original document representation of n_a. Recall that in Section 2.2.2 we present a method for state representation $\boldsymbol{\mu}_t$. Here, we use the same method to encode an anchor graph into a latent vector, but use a different set of parameters since it is used for recommendation purpose rather than for actor-critic learning. We denote the state representation with $\hat{\boldsymbol{\mu}}_T$, which will be used to refine the document representation:

$$\widehat{\boldsymbol{v}}_a = \mathcal{H}_\psi(\boldsymbol{v}_a, \mathcal{X}_a) = Tanh\left(W_5 ELU(W_6(\boldsymbol{v}_a \oplus \hat{\boldsymbol{\mu}}_T))\right) \tag{15}$$

The closeness of two articles, n_a and n_b, is measured as the cosine similarity between their knowledge-aware representations:

$$Sim(\widehat{\boldsymbol{v}}_a, \widehat{\boldsymbol{v}}_b) = Cosine(\widehat{\boldsymbol{v}}_a, \widehat{\boldsymbol{v}}_b) \tag{16}$$

$Sim(\widehat{\boldsymbol{v}}_a, \widehat{\boldsymbol{v}}_b)$ will be the prime indicator of the similarity relationship between an item pair. Meanwhile, we hope the anchor graph itself, to some extent, can be used to judge the relationship between an item pair. To this end, we design an auxiliary scoring model to predict the label of an item pair with the path score $score_\psi$:

$$score_\psi(\mathcal{X}_a, \mathcal{X}_b) = \sum_{p_i \in \Delta_{a \Leftrightarrow b}} \sigma\left(W_7 * ReLU(W_8 * GRU_{path}(p_i))\right) \tag{17}$$

where σ is the *sigmoid* function, $\Delta_{a \Leftrightarrow b}$ is the collection of connecting paths between anchor graph $\mathcal{X}_a$ and $\mathcal{X}_b$. $GRU_{path}(p_i)$ is a sequential path encoder with the Gated Recurrent Unit (GRU) [4] to encode the path $p_i = (\boldsymbol{r}_1 + \boldsymbol{e}_1, \boldsymbol{r}_2 + \boldsymbol{e}_2, \cdot + \cdot, \boldsymbol{r}_{|p_i|} + \boldsymbol{e}_{|p_i|})$ to a latent vector. W_7, W_8 are learnable parameters. We hope the $score_\psi$ of positive item pairs to approach 1 and the $score_\psi$ of negative item pairs to approach 0.

2.4 The Optimization Framework

In this section, we introduce the multi-task optimization framework to train our AnchorKG. An overview of the training pipeline is illustrated in Algorithm 1 and Figure 5 in the Appendix. We start by introducing the actor-critic based reinforcement learning framework to train the anchor graph generator.

2.4.1 Critic. The critic evaluates an action by estimating the action-value function $Q(s, a)$ in the MDP environment. For the critic network, we use a similar structure to the actor network (Eq.8), but with a different set of parameters W_9 and W_{10} to be learned:

$$Q_\phi(s_t, a_i) = W_9\ ReLU\left(W_{10}(\boldsymbol{s}_t \oplus \boldsymbol{a}_i)\right) \tag{18}$$

The critic network is trained with the Temporal Difference (TD) method [10], which learns model parameters by bootstrapping from the current estimate of the action-value function. It first calculates a target q_t according to the Bellman equation[1]:

$$q_t = \mathcal{R}(t) + \mathbb{E}_{a \sim \pi_\theta}\left[\gamma \cdot Q_\phi(s_{t+1}, a)\right] \tag{19}$$

where γ is a decay factor. Interestingly, in experiments we find that setting it to 1.0 leads to the best performance. $\mathcal{R}(t)$ is a compound reward:

$$\mathcal{R}(t) = \alpha_2 \cdot \mathcal{R}_I(t) + (1 - \alpha_2) \cdot \mathcal{R}_T(t) \tag{20}$$

where $\alpha_2 \in [0, 1]$ is a weighting coefficient for coordinating the importance of terminate reward and immediate reward. The critic network is learned by minimizing the TD error:

$$\mathcal{L}_{critic}(\phi) = (Q_\phi(s_t, a_t) - q_t)^2 \tag{21}$$

2.4.2 Actor. The actor aims to maximize the expected payoff of each step w.r.t. the given Q-function:

$$J_{actor}(\theta) = \mathbb{E}_{a \sim \pi_\theta}\left[Q_\phi(s_t, a)\right] \tag{22}$$

We use the policy gradient method [12] to optimize the actor network. The gradients of $J(\theta)\ w.r.t.\ \theta$ are calculated as follows for each sampled trajectory:

$$\nabla_\theta J_{actor}(\theta) \simeq Q_\phi(s_t, a_t) \cdot \nabla_\theta log \pi_\theta(s_t, a_t) \tag{23}$$

We omit the detailed formula derivation of Eq.23, as it is quite straightforward based on existing literature. Interested readers may refer to this helpful tutorial[2] for more comprehensive introduction.

2.4.3 Warm Start Stage. If directly applying the actor-critic reinforcement learning method to train the AnchorKG model from random initialized parameters, the model will show very poor convergence properties [23, 25], due to the uncertainty from the actor and the critic simultaneously, as well as the huge action space from the complex generic knowledge graph. Thus, we adopt the idea of behavior cloning [8], to guide the model training from expert demonstrations as a warm start training process. Expert demonstrations are extracted by running KPRN [18] on a small warm-up dataset, in which for each positive item pair we sample one negative item pair. Note that due to the complexity of KPRN, it is not feasible to serve the candidate recall task. So we compose the small dataset to train KPRN as a ranking task. After KPRN converges, it can produce some high-quality paths which can serve for reasoning. We take these paths as expert demonstrations and encourage the policy network to generate paths identical to them. Note that expert demonstrations extracted by KPRN are imperfect, they are incomplete and may contain noise. This supervised learning stage can result in a well-initialized policy network, which helps the AnchorKG model converge to a satisfying solution after continuing the RL training process.

2.4.4 Negative Sampling. Traditional negative instances sampler, such as popularity-biased sampling [3] or dynamic negative sampling [24], are not informative for optimizing the AnchorKG framework. The reason is that, due to the large scale of the knowledge graph and the vast amount of news articles, it is hard for two randomly sampled news articles to have overlapping entities over the knowledge graph. Some reward signals in our framework will severally suffer from the quality of negative instances. For example, suppose there is a negative item pair (n_a, n_j), if n_a and n_j locate far away from the knowledge graph, there will definitely be no

[1] https://en.wikipedia.org/wiki/Bellman_equation

[2] https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html

overlap between their corresponding anchor graph. Thus, Eq.10 will downgrade to:

$$\hat{\mathcal{R}}_T^{(2)} = \mathbb{E}_{i\sim\kappa(a)}\left[\tanh(\frac{|\mathcal{E}_a \cap \mathcal{E}_i|}{log(|\mathcal{E}_a| \cdot |\mathcal{E}_i|)}) + \tanh(\frac{score_q}{log(|\mathcal{E}_a| \cdot |\mathcal{E}_i|)})\right]$$

Without the supervision of the negative component, the policy network is prone to covering trivial entities, such as popular entities like *the United States*. Thus, we design a simple but effective knowledge-aware negative sampling strategy. We first randomly sample a connectivity number $k \in [1, 2, 3]$, then conduct a random walk along a k-hop path starting from the virtue article node e_{n_a}, suppose the ending node is e_k. Then we will uniformly sample one article n_b from $\Gamma(e_k)$ as the negative candidate if n_b does not belong to the positive candidate set of n_a.

2.4.5 Multi-Task Learning. There are two tasks in our framework, i.e. the anchor graph generation task (generator) and the item2item recommendation task (recommender). These two tasks interact with each other closely, with the generator providing knowledge-aware signals for enhancing the recommender, and the recommender providing reward signals to help optimize the generator. The learning objectives for the generator task are already discussed with Eq.21 and Eq.22. For the recommender task, learning objectives are comprised of two views, i.e., an embedding-based recommendation view and a reasoning-based recommendation view. On the one hand, we rely on the enhanced document representation $\boldsymbol{v}$ to make precise recommendations. Thus the training process is to maximize the probability of positive candidate from a list of candidates mixed with negative ones:

$$P(j|i) = \frac{exp(\alpha \cdot Sim(\widehat{\boldsymbol{v}}_i, \widehat{\boldsymbol{v}}_j))}{\sum_{j' \in \eta(i)} exp(\alpha \cdot Sim(\widehat{\boldsymbol{v}}_i, \widehat{\boldsymbol{v}}_{j'}))} \tag{24}$$

$$\mathcal{L}_{rec}(\psi) = -log \prod_{(i,j)\in O^+} P(j|i) \tag{25}$$

where, α is a smoothing factor for the softmax function, which is set to 10. O^+ denotes the set of positive item pairs. $Sim(\cdot)$ is defined in Eq.16, $\eta(i)$ indicates the set of candidates item for n_i, in which only n_j is positive item and all the rest are negative items. On the other hand, we hope the reasoning paths from anchor graphs can distinguish positive instances from negative instances. To this end, we further pose a binary cross-entropy loss:

$$\begin{aligned}\mathcal{L}_{reason}(\psi) = &- \sum_{(i,j)\in O^+} log\left(score_\psi(\mathcal{X}_i, \mathcal{X}_j)\right) \\ &- \sum_{(i,j)\in O^-} log\left(1 - score_\psi(\mathcal{X}_i, \mathcal{X}_j)\right)\end{aligned} \tag{26}$$

where $score_q(\mathcal{X}_i, \mathcal{X}_j)$ is defined in Eq.17. Similar to [19], we adopt an iterative optimization to train the generator task and the recommender task alternately. At each iteration, we first freeze the recommender parameters (ψ) and optimize the generator according to Eq.21 and Eq.22, then freeze the generator parameters (θ and ϕ) and optimize the recommender according to Eq.25 and Eq.26. The iteration goes on until the model converges.

3 EXPERIMENTS

3.1 Experiment Settings

We conduct experiments on two datasets: *MIND*[3] [21] and *Bing News*[4]. *MIND* is the largest open-source English news dataset for public research purpose, and *Bing News* is a private dataset extracted from Bing news service. We use Wikidata as the knowledge graph for both datasets. After data cleaning, we get 410,268 positive instances for MIND and 64,837 positive instances for the Bing News dataset. The details about data processing step as well as the statistics of the datasets can be found at A.1 in the Appendix. The positive instances are split into training/validation/test set by 80%/10%/10%. For the training/validation set, we sample 4 negative instances for each positive instance. For the test set, we don't sample negative instances, the goal is to discover the positive items from the entire set of item candidates.

We compare AnchorKG with two groups of models:

- **DKN** [14], **NAML** [20], **SentenceBERT** [9], **KRED** [6] are four strong baselines which fully leverage textual content for news recommendations. But they cannot provide reasoning.
- **PGPR** [22], **ADAC** [25] are two strong baselines with knowledge-aware reasoning ability. However, news contents are not sophisticated modeled when compared to the first groups of baselines.

Considering that PGPR and ADAC can only be used in the recall stage of recommender systems, in this section, we report experiments based on the recall setting, i.e., given a source article a, the task is to target the related articles $\{b\}$ from the entire space of article candidates. We adapt DKN, NAML, and KRED models slightly to make them fit for this scenario. The relationship between two items is measured as the cosine similarity of their latent representations, so we only need the text encoder component from these models, and the models are all optimized by the softmax loss with Eq.25. Hyper-parameters are reported in Appendix A.2.

We use the same top-N recommendation metrics as used in [22, 25]: Precision (Prec.), Recall, Normalized Discounted Cumulative Gain (NDCG), and Hit Ratio (HR).

3.2 Overall Performance

Table 1 reports the top-N recommendation performance of different models. We make the following observations:

- Embedding-based baselines, including DKN, NAML and KRED, are designed for news recommendations with strong text encoders. They substantially outperform both PGPR and ADAC across different metrics, verifying that a deep understanding of news content is critical for recommendation performance. However, this group of methods cannot produce explainable reasons for recommendations.
- ADAC and PGPR are path-finding-based methods for recommendations and are recognized as state-of-the-art recommendation reasoning methods over knowledge graphs. ADAC leverages demonstrations to enhance the pathfinding strategy in the reinforcement learning framework, so its performance is slightly better than PGPR, which matches the conclusions in [25]. Despite their ability for reasoning, their major weakness lies in the low

[3] https://msnews.github.io/index.html
[4] https://www.bing.com/news

Table 1: Overall recommendation performance of different models on two datasets. Results are reported in percentage (%).

	MIND Dataset								Bing News Dataset							
	Top-5				Top-10				Top-5				Top-10			
	NDCG	Prec.	Recall	HR	NDCG	Prec.	Recall	HR	NDCG	Prec.	Recall	HR	NDCG	Prec.	Recall	HR
DKN	3.428	2.217	6.252	8.145	5.258	1.596	10.23	13.41	2.929	1.467	3.466	6.099	4.225	1.001	4.978	9.281
NAML	3.718	2.311	6.879	8.712	5.706	1.734	11.09	14.54	3.318	1.727	4.192	7.357	5.109	1.186	6.063	11.20
S-BERT	3.929	2.426	6.929	9.285	5.997	1.820	11.67	15.30	3.455	1.789	4.263	7.560	5.227	1.224	6.108	11.56
KRED	4.036	2.591	7.157	9.603	6.106	1.869	12.20	16.00	3.617	1.878	4.474	7.936	5.450	1.286	6.813	11.95
PGPR	2.484	1.618	3.894	5.731	4.130	1.229	6.701	9.898	1.752	0.983	1.851	3.306	2.168	0.624	2.630	4.851
ADAC	2.671	1.650	4.110	5.855	4.295	1.276	6.893	10.09	1.777	1.002	1.886	3.371	2.216	0.648	2.693	4.998
AnchorKG	**4.301**	**2.805**	**8.186**	**10.755**	**6.621**	**2.112**	**14.71**	**17.92**	**3.866**	**2.046**	**4.733**	**8.370**	**5.623**	**1.497**	**7.513**	**12.49**

accuracy of news recommendations, where the textual content plays a key role in matching news relevance.

- AnchorKG combines the advantage of embedding-based methods and path-find-based methods. It outperforms all the baselines in terms of different metrics, and the conclusions are consistent on two datasets, which demonstrates the effectiveness of AnchorKG. Specifically, compared with the first group of baselines, AnchorKG is not only more accurate (which is achieved by fusing an anchor knowledge graph into a strong text encoder SentenceBERT), but also able to give knowledge paths as explainable reasons; compared with the second group of baselines, AnchorKG is remarkably more accurate. We will further demonstrate the reasoning performance comparison in Table 3.

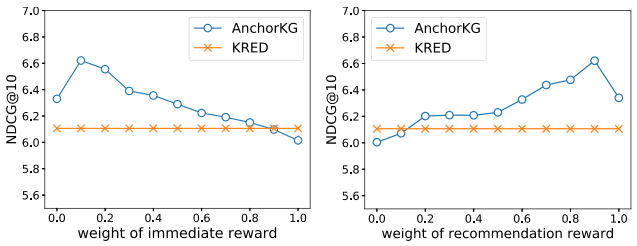
3.3 Ablation and Hyper-Parameters Study

We study the necessity of key components, including the knowledge-aware negative sampler, the warm-up learning stage, and the reward signals. For the knowledge-aware negative sampler, we replace it with a popularity-based negative sampler (marked as *w/o KG negative sampler*). For the warm-up learning component, we compare an AnchorKG model without enabling this stage (denoted as *w/o KG negative sampler*). From Table 2 we can observe that changing either of these two components will lead to a performance drop, which demonstrates the effectiveness of these two components.

As for the influence of reward signals, key ingredients include the immediate reward & terminal reward, and the recommendation reward & reasoning reward. These ingredients are linearly compounded together, so we examine their effectiveness by varying the corresponding weighting coefficients, i.e. α_1 in Eq.11 and α_2 in Eq.20. The results are reported in Figure 2a and Figure 2b. We observe that a proper setting for α_2 is 0.1, and that for α_1 is 0.9.

Table 2: Ablation study on removing the knowledge-aware negative sampler or the warm-up learning stage. Results are reported in percentage (%) based on top-10.

	NDCG	Prec.	Recall	HR
	MIND Dataset			
AnchorKG	**6.621**	**2.112**	**14.71**	**17.92**
w/o KG negative sampler	6.462	2.028	14.45	17.47
w/o warm-up learning	6.287	2.006	14.06	17.12
	Bing News Dataset			
AnchorKG	**5.623**	**1.497**	**7.513**	**12.49**
w/o KG negative sampler	5.442	1.302	6.987	11.82
w/o warm-up learning	5.454	1.383	7.086	12.11



(a) α_2 in Eq.20 **(b) α_1 in Eq.11**

Figure 2: A study of the influence of the immediate reward (left) and the terminal reward (right) on MIND, by varying α_2 in Eq.20 and α_1 in Eq.11, respectively.

3.4 The Quality of Reasoning

Next, we study the reasoning ability of AnchorKG. As revealed in [22, 25], designing quantitative metrics for evaluating the quality of explainable reasoning is very challenging and subjective. In this section, besides case studies, we advocate using two metrics for quantitative analysis. The first measure is average number of connecting paths for positive item pairs. The second measure is the GRU path score from Eq.17. Recall that to encourage the model to pick up high-quality paths, we design a GRU-based path encoder to score every path. Results can be found in Table 3. PGPR and ADAC use a single path for reasoning, so their accumulated path scores

Table 3: The quantitative analysis of reasoning capability.

Method	Avg. #. Connect Paths	Sum of Quality Score	Max Single Path Quality Score
	MIND Dataset		
PGPR	1	0.626	0.626
ADAC	1	0.651	0.651
AnchorKG	**5.29**	**3.769**	**0.807**
	Bing News Dataset		
PGPR	1	0.589	0.589
ADAC	1	0.593	0.593
AnchorKG	**3.55**	**2.173**	**0.796**

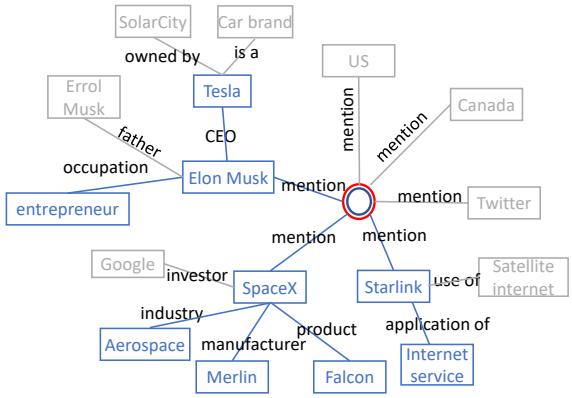


Figure 3: The generated anchor knowledge graph for one piece of case study news from the MIND dataset.

are equivalent to their maximum single path scores. On average, AnchorKG can discover 5.29 paths on MIND and 3.55 paths on Bing News dataset for each positive item pair. What is more, the average maximum single path score is also significantly higher.

To illustrate how AnchorKG generates a subgraph over the knowledge graph, we offer a real news article example from MIND:

Elon Musk sends a tweet through SpaceX's Starlink broadband satellite. The company reportedly hopes to bring Starlink broadband services in the US and Canada by the middle of next year. As SpaceX pushes to take over the night sky with 30,000 Starlink satellites, CEO Elon Musk tested the orbiting routers' internet connectivity early Tuesday with a message to his nearly 29 million Twitter followers. "Sending this tweet through space via Starlink satellite," he wrote, before following up to express his surprise. "Whoa, it worked!!"

Entities in the article are highlighted with underline. Due to the space limit, we can only draw part of the anchor graph in Figure 3. From Figure 3, it is straightforward to see that the generated anchor graph is biased towards the *SpaceX* related topics. Entities such as *US*, *Canada* and *Twitter* is ignored due to their irrelevance, even though they are directly connected to the news node. *Elon Musk* owns dual identities, i.e. the CEO of *Tesla* and the founder of *SpaceX*. However, for this news article, the topic is only related to *SpaceX*. We observe that the generated anchor graph indeed covers more entities related to *SpaceX*, with some flaw that a few general entities, such as *Internet service*, are also included.

Figure 4 demonstrates a knowledge-aware reasoning result for a typical pair of news items related to sports. The most important single path is $path\ 1: n_a \xrightarrow{mention} ATP\ Finals \xleftarrow{mention} n_b$, which indicates that both articles focus on reporting *ATP Finals*. However, considering that there are so many articles which have mentioned *ATP Finals*, a single reasoning path may not be sufficiently convincing. Thanks to the anchor graphs, we can generate multiple reasoning paths, such as $path\ 2: n_a \xrightarrow{mention} Nadal \xrightarrow{participate\ in} US\ Open \xleftarrow{participate\ in} Berrettini \xleftarrow{mention} n_b$, and $path\ 3: n_a \xrightarrow{mention} Nadal \xrightarrow{country} Spain \xleftarrow{country} Carreno\ Busta \xleftarrow{mention} n_b$, and so on, which strongly enhance the evidence.

4 RELATED WORK

4.1 KGs for Recommendation Accuracy

Knowledge graphs have been widely used to improve the recommendation accuracy. [13] proposes RippleNet, which stimulates the propagation of users' preferences over knowledge entities by expanding users' potential interests along links in the knowledge graph iteratively. [16] develops an end-to-end multi-task training framework (MKR), with the goal of utilizing knowledge graph embedding to assist the recommendation task. The key component of MKR is the cross & compress units, which can share latent features and learn high-order interactions between items in recommender systems and entities in the knowledge graph. Similarly, [2] also considers the joint learning of recommendation and knowledge graph completion. With the emerging techniques of graph neural networks (GNN), some researchers devise GNN-based models to utilize knowledge graphs for recommendation systems [15, 17]. As for news recommendation domain, using knowledge graphs to enhance the representation of news articles is an effective approach. [14] proposes a knowledge-aware convolutional neural network (KCNN) to fuse semantic-level and knowledge-level representations of news articles. [6] introduces KRED, which leverages KGs to enhance an arbitrary type of article representation. However, all these works focus on improving the accuracy performance of recommender systems. In this paper, we focus more on recommendation reasoning with knowledge graphs.

4.2 KGs for Recommendation Reasoning

Another major merit of knowledge graphs is that they can endow recommender systems with explainability ability. Connectivity between two nodes with a multi-hop path over the graph can serve as knowledge-aware *reasoning*, because it reveals the semantic relationship between two nodes. [18] searches all potential paths connecting the user and the item, then adopt an LSTM on paths to capture the sequential dependencies of nodes for user preference inference. Reasoning is conducted by selecting the paths with highest preference scores. To avoid enumerating all possible paths, [22] proposes a reinforced method called Policy-Guided Path Reasoning (PGPR), with three key components including a soft reward strategy, a user-conditional action pruning strategy, and a multi-hop scoring approach. [25] argues that the path-finding process in previous methods will result in issues with respect to convergence and explainability, so the authors design a demonstration-based knowledge graph reasoning framework, with a key component of Adversarial Actor-Critic (ADAC) model. OUr paper is most related to this line of research, we have pointed out the motivation and difference between this paper and existing works in Section 1.

5 CONCLUSIONS

In this paper, we propose a novel knowledge-aware reasoning paradigm, AnchorKG, for news recommendation and reasoning. AnchorKG generates a compact anchor knowledge graph for each article individually, and the relationship reasoning between any two articles can be efficiently conducted by interactions between their anchor graphs. We formulate the anchor graphs generation as a Markov Decision Process and solve it with reinforcement learning

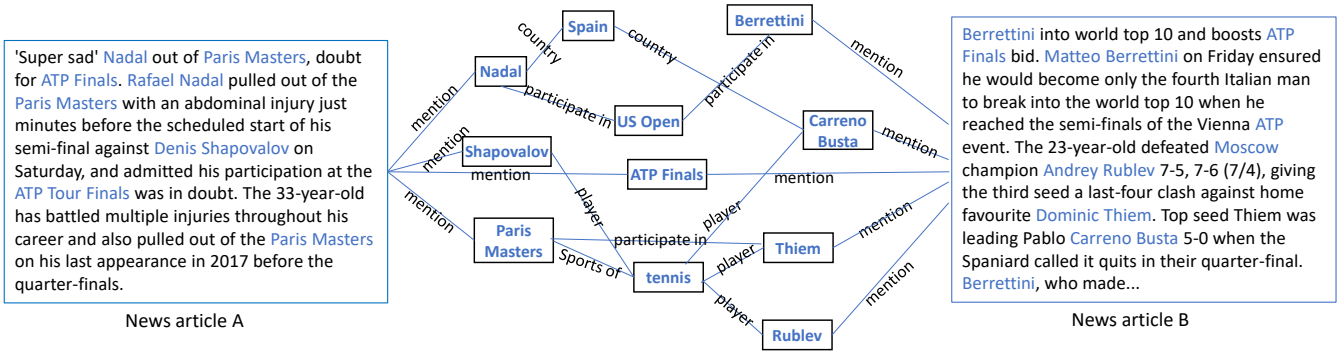


Figure 4: An example of reasoning results between two news articles based on their anchor graphs on MIND.

techniques, without the demand of enumerating all the possible connecting k-hop paths. The AnchorKG paradigm possesses a few important merits. First, the generated anchor graph can be used to enhance the representation of news articles, so the accuracy of recommendations can be improved. Second, the interactions between the corresponding anchor graphs of two articles can naturally serve as reasoning paths for explainability. Third, the generation process of anchor graph does not require any contextual information (such as a given target candidate). Therefore, once the offline anchor graphs are cached, our model is practical for efficient online serving. Extensive experiments on one public dataset and one industrial dataset demonstrate the effectiveness of AnchorKG. For future work, we plan to conduct experiments with the AnchorKG paradigm on the user-to-item recommendation scenario.

REFERENCES

[1] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS'13* (Lake Tahoe, Nevada). Curran Associates Inc., USA, 2787–2795.

[2] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. [n.d.]. Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*. 151–161.

[3] Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. 2017. On sampling strategies for neural network-based collaborative filtering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[4] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Association for Computational Linguistics, Doha, Qatar, 103–111. https://doi.org/10.3115/v1/W14-4012

[5] Danyang Liu, Ting Bai, Jianxun Lian, Xin Zhao, Guangzhong Sun, Ji-Rong Wen, and Xing Xie. 2019. News Graph: An Enhanced Knowledge Graph for News Recommendation. In *KaRS@CIKM 2019, Beijing, China, November 7, 2019*. 1–7.

[6] Danyang Liu, Jianxun Lian, Shiyin Wang, Ying Qiao, Jiun-Hung Chen, Guangzhong Sun, and Xing Xie. 2020. KRED: Knowledge-Aware Document Representation for News Recommendations. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20)*. New York, NY, USA, 200–209.

[7] Ting Liu, Andrew W Moore, Ke Yang, and Alexander G Gray. 2005. An investigation of practical approximate nearest neighbor algorithms. In *Advances in neural information processing systems*. 825–832.

[8] Dean A Pomerleau. 1991. Efficient training of artificial neural networks for autonomous navigation. *Neural computation* 3, 1 (1991), 88–97.

[9] Nils Reimers and Iryna Gurevych. [n.d.]. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 3982–3992.

[10] Richard S Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine learning* 3, 1 (1988), 9–44.

[11] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[12] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*. 1057–1063.

[13] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*. ACM, 417–426.

[14] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep Knowledge-Aware Network for News Recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. ACM, 1835–1844.

[15] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. [n.d.]. Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. 968–977.

[16] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*. ACM, 2000–2010.

[17] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. ACM, 950–958.

[18] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5329–5336.

[19] Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. 2020. Reinforced Negative Sampling over Knowledge Graph for Recommendation. In *Proceedings of The Web Conference 2020* (Taipei, Taiwan) *(WWW '20)*. Association for Computing Machinery, 99–109.

[20] Chuhan Wu, Fangzhao Wu, Mingxiao An, Jianqiang Huang, Yongfeng Huang, and Xing Xie. [n.d.]. Neural News Recommendation with Attentive Multi-View Learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. 3863–3869.

[21] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, and Ming Zhou. 2020. MIND: A Large-scale Dataset for News Recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 3597–3606.

[22] Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 285–294.

[23] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690* (2017).

[24] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 785–788.

[25] Kangzhi Zhao, Xiting Wang, Yuren Zhang, Li Zhao, Zheng Liu, Chunxiao Xing, and Xing Xie. 2020. Leveraging Demonstrations for Reinforcement Recommendation Reasoning over Knowledge Graphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

A APPENDIX

For a better understanding of our work, we provide a demo at https://www.youtube.com/watch?v=B8vOVZ1ZYr4.

A.1 Dataset Details

MIND is the largest open-source English news recommendation dataset for public research purpose, constructed from the user click logs of Microsoft News. The Bing News dataset contains one month's impression logs extracted from a news reading service of our company. The original MIND dataset contains 1 million users and more than 161k news articles, and the Bing News dataset contains 141k users and 1.7 million news articles. In this paper we consider the item-to-item recommendation scenario, so we convert the dataset into relationships between items. Each instance can be formulated as a triple $(a, b, y_{a,b})$, where $y_{a,b} \in [0, 1]$ is the label indicating whether the item pair (a, b) is related or not. Positive label is determined by a co-click score, which is defined as $\frac{\#.users\ click\ both\ a\ and\ b}{\sqrt{\#.users\ click\ on\ a} \times \sqrt{\#.users\ click\ on\ b}}$. We manually compare the article pairs with different co-click scores, and empirically set a threshold of 0.05. As long as the co-click score is greater than this threshold, the pair of articles is judged as a positive instance. Through this setting, we get 410,268 pairs of positive instances from the MIND dataset and 64,837 positive instances from the Bing News dataset. The basic statistics of our experimental dataset is depicted in Table 4. We use Wikidata [5] as the the knowledge graph

Table 4: Statistics of the item2item news datasets after data cleaning.

	MIND Dataset	Bing News Dataset
# . articles	161,013	31,991
#. positive pairs	410,268	64,837
#. entities per title	1.2	1.1
#. entities per article	16	3.4
#. words per article	639	58
#. unique entities	205,223	16,487

for experiments. Wikidata is a free and open knowledge base, we download the whole graph from its storage page[6]. Considering that the original Wikidata is a generic knowledge graph which may contain a lot of news-irrelevant entities and relations, we do some graph pruning to select a subgraph from Wikidata, and add a group of topic nodes to the knowledge graph, following the techniques introduce in News Graph [5]. The MIND dataset provides entities mentioned in articles, and their IDs can be joined with Wikidata's entity IDs. For the Bing News dataset, we have a commercial NER toolkit to link knowledge entities to the Wikidata. We first gather all the entities from the dataset, then extend their 1-hop connecting entities in Wikidata. Next, we collect all the relations connecting these entities. After that, we will get a subgraph from Wikidata, which will be used throughout our experiments. The basic statistics of knowledge graphs can be found in Table 5.

[5]https://www.wikidata.org/wiki/Wikidata:Main_Page
[6]https://dumps.wikimedia.org/wikidatawiki/entities/

Table 5: Statistics of the knowledge graph.

	the original Wikidata	Wikidata after pruning
#. entities	67,719,428	3,275,149
#. triples	413,467,881	31,963,632
#. relations	6,896	1,091

A.2 Hyper-Parameter Settings

Thanks to the authors of related papers, all the baseline methods are open-source and can be found at GitHub. We adopt their original implementations and make some minor changes for our experiments. The embedding size for entities and relations in TransE is 128. For NAML, the number of CNN filters is 400 and windows size is 3. For DKN, the number of CNN filters is 100 and windows sizes are [1,2,3]. For KRED and AnchorKG, we use SentenceBERT [9] as the basic document encoder, because it modifies the BERT network and uses Siamese network structures to derive semantically meaningful embeddings that can be compared using cosine similarity. For all models, the final document embedding size is set to 128. The maximum path length of PGPR and ADAC is set to 3 as suggested in their papers. For a fair comparison, the maximum depth of AnchorKG is also set to 3. Intermediate parameters W_* in AnchorKG are also set to 128 series, e.g., $\mathbb{R}^{1\times128}$ or $\mathbb{R}^{128\times128}$.

A.3 Impact of Receptive Field Size

The receptive field size is introduced in Section 2.2.2. It is intuitive that if the size is set too large, it will be hard for model to converge due to the exponential growth of the scale and meanwhile too much noisy signals will be absorbed in. On the other hand, if the size if set too small, useful information will inevitably be excluded. From Table 6 we can see that a proper setting will be [5, 3, 2].

A.4 A Graphical Illustration of AnchorKG

To help readers get an intuitive sense of the framework, we provide a graphical illustration of the model in Figure 5 as well as a pseudo-code in Algorithm 1.

Table 6: A study on impacts of the receptive field sizes for anchor graphs generation on the MIND dataset. Results are reported in percentage (%) at top-10.

Sizes	NDCG	Prec.	Recall	HR
[5, 5, 5]	6.356	2.060	14.24	17.38
[5, 5, 1]	6.523	2.091	14.49	17.66
[5, 3, 3]	6.568	2.093	14.53	17.66
[5, 3, 2]	**6.621**	**2.112**	**14.71**	**17.92**
[3, 3, 2]	6.291	2.017	14.05	16.89
[3, 2, 1]	6.190	1.955	13.75	16.36

A.5 Discussions about Online Serving

We briefly discuss how AnchorKG is feasible for online serving. Since the generation of anchor graphs is individually independent, we can finish the anchor graph generation process for all articles

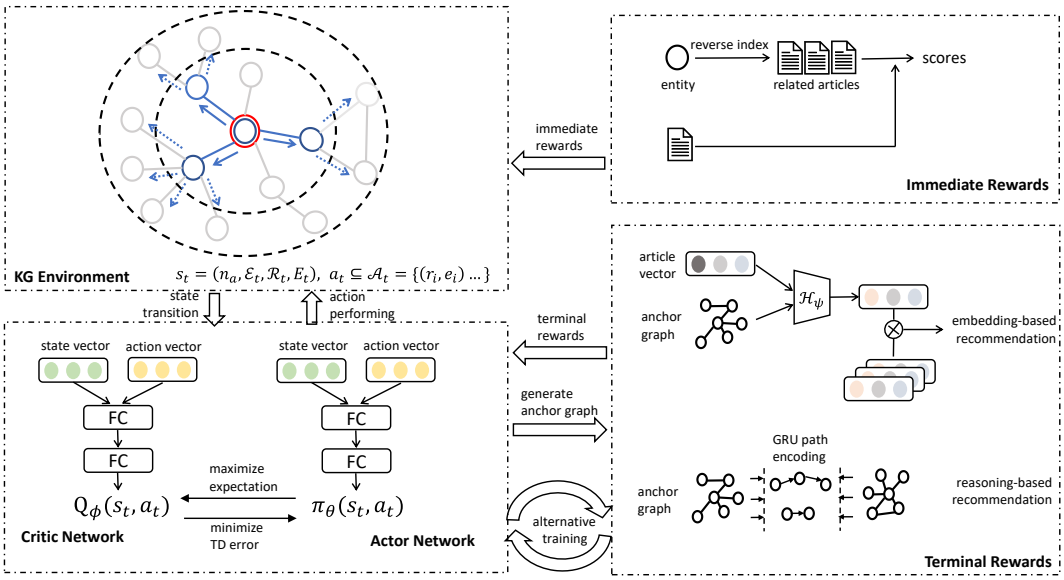


Figure 5: An overview of our proposed AnchorKG, which is formulated as a reinforcement learning framework with a Markov Decision Process. An agent interacts with the Knowledge Graph (KG) environment, and get two types of rewards, i.e. the immediate reward and the terminal reward.

Algorithm 1 AnchorKG Training Pipeline

Input: positive news pairs set: $\mathcal{N}$; base embeddings $\mathcal{V}$ of news; knowledge graph $\mathcal{G}$;

Output: AnchorKG generator: π_θ; news enhancement module: $\mathcal{H}(\psi)$; reasoning module: $score(\psi)$;

Randomly initialize all parameters;
Warm up training of generator parameters in $\{Q_\phi, \pi_\theta\}$ ▷ Section 2.4.3;
repeat
  for each pair (n_1, n_2) in $\mathcal{N}$ **do**
    Perform negative sampling for (n_1, n_2) ▷ Section 2.4.4
    Generate anchor graphs $\mathcal{X}$ with π_θ ▷ Section 2.2.2;
    Calculate immediate rewards: R_I with Eq.(14)
    Refine the document representation: $\widetilde{\boldsymbol{v}}$ with Eq.(15);
    Calculate recommendation reward: $\mathcal{R}_T^{(1)}$ with Eq.(9)
    calculate reasoning reward: $\mathcal{R}_T^{(2)}$ with Eq.(10)
    calculate final reward $\mathcal{R}(t)$ with Eq.(20)
    Update generator-related modules $\{Q_\phi, \pi_\theta\}$ with Eq.(21) and Eq.(22)
    Update $\mathcal{H}(\psi)$ with Eq.(25);
    Update $score(\psi)$ with to Eq.(26);
  end for
until converge

in a offline distributed inference manner, enhance articles representation vectors with the anchor graphs, and then cache the results. In the recall stage, positive candidates can be retrieved by an approximate nearest neighbor searching (ANN) [7] module, based on the enhanced article representation vectors. The reasoning paths of any given item pairs (which is a typical ranking scenario) can be derived from the interactions of the corresponding anchor graph, which should be fast due to the compact structure of anchor graphs (usually less than 50 entities).