Empowering recommender systems using automatically generated Knowledge Graphs and Reinforcement Learning

Ghanshyam Verma*
ghanshyam.verma@insight-centre.org
University of Galway
Ireland

Simon Simanta
simon.simanta@insight-centre.org
University of Galway
Ireland

Huan Chen
huan.chen@insight-centre.org
University of Galway
Ireland

Devishree Pillai
devishree.pillai@insight-centre.org
University of Galway
Ireland

John P. McCrae
john.mccrae@universityofgalway.ie
University of Galway
Ireland

János A. Perge
Janos.Perge@fmr.com
Fidelity Investments
USA

Shovon Sengupta
Shovon.Sengupta@fmr.com
Fidelity Investments
USA

Paul Buitelaar
paul.buitelaar@universityofgalway.ie
University of Galway
Ireland

ABSTRACT

Personalized recommendations have a growing importance in direct marketing, which motivates research to enhance customer experiences by knowledge graph (KG) applications. For example, in financial services, companies may benefit from providing relevant financial articles to their customers to cultivate relationships, foster client engagement and promote informed financial decisions. While several approaches center on KG-based recommender systems for improved content, in this study we focus on interpretable KG-based recommender systems for decision-making.

To this end, we present two knowledge graph-based approaches for personalized article recommendations for a set of customers of a large multinational financial services company. The first approach employs Reinforcement Learning (RL) and the second approach uses the XGBoost algorithm for recommending articles to the customers. Both approaches make use of a KG generated from both structured (tabular data) and unstructured data (a large body of text data).

Using the RL-based recommender system we could leverage the graph traversal path leading to the recommendation as a way to generate interpretations (Path Directed Reasoning (PDR)). In the XGBoost-based approach, one can also provide explainable results using post-hoc methods such as SHAP (SHapley Additive exPlanations) and ELI5 (Explain Like I'm Five).

We also compared the above approaches with published algorithms for building recommender systems. Our proposed RL-based recommender system achieved 43.76% MAP (MAP@K=10). Our RL-based recommender system outperformed both the XGBoost-based approach and baseline model (Bayesian personalized ranking) by 13.38 and by 32.55 percentage points, respectively, delivering more accurate and personalized article recommendations. Importantly, our approach offers explainable results, promoting better decision-making. This study underscores the potential of combining advanced machine learning techniques with KG-driven insights to bolster experience in customer relationship management.

CCS CONCEPTS

• **Artificial Intelligence** → **Machine Learning**; *Recommender Systems*; Reinforcement Learning; XGBoost; • **Natural Language Processing** → Knowledge Graph.

KEYWORDS

Knowledge Graph Embedding, TuckER, TransE, XAI

ACM Reference Format:
Ghanshyam Verma, Simon Simanta, Huan Chen, Devishree Pillai, John P. McCrae, János A. Perge, Shovon Sengupta, and Paul Buitelaar. 2023. Empowering recommender systems using automatically generated Knowledge Graphs and Reinforcement Learning. In *Proceedings of online and adaptive recommender systems (KDD-OARS).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The increasing demand for personalized content has led to the development of recommendation systems that can effectively utilize structured information. Knowledge graphs (KGs) have emerged as a promising solution for this challenge, offering improved recommendation performance and explainability due to the inherent comprehensibility of relationships between entities [24]. A growing body of research is dedicated to exploring the potential of knowledge graph reasoning in personalized recommendation [1, 4, 10, 33]. One line of research focuses on knowledge graph embedding models, such as TransE [4] and node2vec [10], which align the knowledge graph in a regularized vector space, identifying the similarity between entities by calculating the distance between their representations [41]. However, purely KG embedding-based approaches struggle to uncover multi-hop relational paths, limiting the ability to capture complex relationships between entities. Another line of research investigates path-based recommendation techniques. Gao et al. [9] proposed the concept of meta-paths for reasoning over KGs. Although promising, this approach faces challenges when dealing with the numerous types of relations and entities present in large, real-world KGs, making it difficult to explore relationships between unconnected entities. Wang et al. [37] developed a path embedding approach for recommendation over KGs that enumerates all qualified paths between every user-item pair, followed by training a sequential RNN model to predict ranking scores for the pairs. While this method improves recommendation performance, it is not feasible to explore all paths for every user-item pair in large-scale KGs due to computational limitations. Recent advances have focused on combining collaborative filtering (CF) with KG embedding techniques to enhance recommendation performance [1, 41]. For example, Ai et al. [1] proposed a method that incorporated a soft matching algorithm to identify explanation paths between users and items. However, this strategy generates explanations post-hoc through empirical similarity matching between user and item embeddings, providing retrospective rationales for the chosen recommendations rather than deriving explanations from the reasoning process. We argue that an intelligent recommendation agent should explicitly reason over knowledge graphs for decision-making rather than simply embedding the graph as latent vectors for similarity matching. In this paper, we treat knowledge graphs as a flexible structure to maintain the agent's knowledge about users, items, other entities, and their relationships. The agent initiates the process with a user and conducts explicit multi-step path reasoning over the graph, discovering suitable items for recommendation. This approach allows for the reasoning process to be easily interpreted, providing causal evidence for the recommended items. Our goal is not only to select a set of candidate items for recommendation but also to provide the corresponding reasoning paths as interpretable evidence for each recommendation. To address the limitations of previous work, we propose an approach that casts the recommendation problem as a deterministic Markov Decision Process (MDP) over the knowledge graph. We employ a Reinforcement Learning (RL) method, wherein an agent begins with a given user and learns to navigate to potential items of interest. The path in the KG then serves as an explanation for why the item should be recommended to the user. This approach presents three main challenges: measuring the correctness of an item for a user, efficiently exploring promising reasoning paths in the graph, and preserving the diversity of both items and paths during exploration. To tackle these challenges, we propose a KG-driven RL-based approach. The benefit of our approach is that it can also work when reviews or ratings of the items are not available and only click information is available to learn the user preferences.

Our experimental results demonstrate that our proposed method consistently outperforms state-of-the-art recommendation techniques, we present qualitative case studies to demonstrate the explainability of our approach, providing insights into the reasoning paths and decision-making processes of the recommendation agent. These case studies showcase the interpretability of our method, allowing users to better understand the rationale behind the recommendations. In summary, our research contributes to the growing body of literature on knowledge graph-based recommendation systems, specifically in the financial domain. By proposing a novel reinforcement learning approach and conducting a comparative study with the XGBoost algorithm, we offer valuable insights into the potential of knowledge graphs for improving the performance and explainability of personalized recommendation systems. Our development of a KG-driven XGBoost recommendation system further demonstrates the versatility and applicability of knowledge graph techniques in the field of recommendation.

By developing a KG-driven XGBoost recommendation system alongside our reinforcement learning approach, we aim to showcase the flexibility and potential of knowledge graph-based techniques in addressing a wide range of recommendation scenarios. Our comparative study between the two approaches not only provides insights into their respective strengths and limitations but also highlights the importance of tailoring recommendation algorithms to specific application contexts and requirements. We have made public the source code of both the proposed approaches via a GitHub link [1].

Our main contributions are as follows: (1) Automatic KG creation using structured and unstructured data. (2) Use of KG for building an XGBoost-based recommender system that can exploit click information. (3) Use of KG for building an RL-based recommender system that can exploit click information. (4) Explainability module that can explain the results.

The rest of the paper is structured as follows. In Section 2, we describe the existing methods for building recommender systems. Section 3 describes the methodology. Section 4 describes the experimental setup. In Section 5, we discuss and compare results in detail. Finally, we conclude in Section 6.

2 RELATED WORK

2.1 Collaborative Filtering

Collaborative Filtering (CF) has been a cornerstone in the development of recommender systems. Early approaches to CF focused on the user-item rating matrix and predicted ratings using user-based [14, 17, 27] or item-based [21, 28] collaborative filtering methods. These approaches calculated similarities between users or items to generate recommendations.

As dimension reduction methods advanced, latent factor models, such as matrix factorization, gained widespread adoption in recommender systems. Prominent techniques include singular value decomposition [18], non-negative matrix factorization [19], and probabilistic matrix factorization [23]. These methods essentially learn a latent factor representation for each user and item to calculate the matching score of user-item pairs.

In recent years, deep learning and neural models have further extended collaborative filtering, leading to two main sub-categories: similarity learning and representation learning. The similarity learning approach adopts relatively simple user/item embeddings (e.g., one-hot vectors) and learns a complex prediction network as a similarity function to compute user-item matching scores [13]. In contrast, the representation learning approach focuses on learning

[1]https://github.com/GhanshyamVerma/Explainable-Recommender-System.

richer user/item representations, while using a simple similarity function (e.g., inner product) for score matching [36, 42].

However, the recommendation results generated by latent factor or latent representation models can be difficult to explain, which has led to a growing interest in explainable recommendation [19, 20]. The challenge of making recommendations more interpretable has driven researchers to explore various techniques and approaches that offer both high-quality recommendations and meaningful explanations for the user-item associations.

In response to the challenges posed by the lack of interpretability in traditional collaborative filtering approaches, researchers have started to explore hybrid recommender systems that combine the benefits of CF methods with other techniques, such as knowledge graph-based methods [12, 41]. These hybrid systems aim to improve the quality of recommendations while also providing more interpretable and explainable results.

Knowledge graphs provide a structured representation of information, making it easier to reason about the relationships between entities and draw meaningful connections. By incorporating knowledge graphs into the recommendation process, researchers can develop systems that offer both high-quality recommendations and interpretable explanations for user-item associations.

The field of collaborative filtering-based recommender systems has seen significant advancements over the years, with a growing emphasis on integrating additional sources of information and enhancing interpretability. The exploration of hybrid systems, such as those that combine collaborative filtering with content-based filtering or knowledge graph-based methods, holds promise for the development of more accurate, personalized, and explainable recommendations.

2.2 Knowledge Graph-driven Recommender Systems

Knowledge Graph-driven Recommender Systems (KGRS) have recently gained attention due to their ability to provide explainable and high-quality recommendations. Researchers have explored different ways to incorporate knowledge graph embeddings into recommender systems to improve recommendation performance and interpretability. One research direction focuses on leveraging knowledge graph embeddings as rich content information to enhance recommendation performance. For example, Zhang et al. [41] utilized knowledge base embeddings to generate user and item representations for recommendation purposes. Huang et al. [15] employed memory networks over knowledge graph entity embeddings for recommendation. Wang et al. [34] proposed a ripple network approach for embedding-guided multi-hop KG-based recommendation, which allows for the exploration of connections between entities in the knowledge graph. Another research direction aims to leverage the entity and path information in the knowledge graph to make explainable decisions. Ai et al. [1] incorporated the learning of knowledge graph embeddings for explainable recommendation, but their explanation paths are essentially post-hoc explanations, as they are generated by soft matching after the corresponding items have been chosen. Wang et al. [37] proposed an RNN-based model to reason over KGs for recommendation. However, this approach requires enumerating all possible paths between each user-item pair for model training and prediction, which can be impractical for large-scale knowledge graphs.

The field of Knowledge Graph-driven Recommender Systems has witnessed significant progress in recent years. Researchers are exploring different approaches to incorporate knowledge graph embeddings and entity relationships to enhance recommendation performance while providing interpretable and explainable results. Future work in this area will likely focus on developing more efficient and scalable methods for reasoning over large-scale knowledge graphs and further improving the quality and explainability of recommendations.

Some researchers have focused on leveraging the structural properties of knowledge graphs to improve recommendation performance. For instance, Wang et al. [36] developed a graph attention network that incorporates both the relational information and entity features in a knowledge graph for recommendation. This approach allows for more accurate and context-aware recommendations by attending to the most relevant relations and entities for a given user-item pair.

In addition to using knowledge graph embeddings, researchers have also explored incorporating external knowledge sources and incorporating user-item interactions into the knowledge graph. Cao et al. [5] proposed a unified framework for incorporating user-item interactions and external knowledge sources into the knowledge graph, which improved the quality of recommendations by capturing the complex interplay between these elements.

Schlichtkrull et al. [29] introduced a relational graph convolutional network (R-GCN) that learns embeddings for both entities and relations in a knowledge graph. This method can be used in a wide range of applications, including recommender systems, by exploiting the rich information present in the knowledge graph structure.

The research area of Knowledge Graph-driven Recommender Systems has experienced significant advancements, with researchers exploring various methods to utilize knowledge graph embeddings, external knowledge sources, and user-item interactions to improve the quality and explainability of recommendations. As more efficient and scalable techniques are developed, KGRS will continue to evolve and provide increasingly accurate, personalized, and explainable recommendations.

2.3 Reinforcement Learning based Recommender Systems

Reinforcement Learning (RL) has garnered considerable interest in the research community, with numerous successful applications in various domains, including recommender systems. Researchers have explored RL-based recommender systems in both non-KG settings and KG settings for a range of tasks.

In non-KG settings, RL has been applied to various types of recommender systems, such as ads recommendation [32], news recommendation [43], and post-hoc explainable recommendation [35]. These applications have demonstrated the potential of RL to adapt to changing user preferences and generate personalized recommendations based on user interactions.

In the context of knowledge graphs, researchers have primarily focused on utilizing RL for tasks such as question-answering (QA).

For instance, Xiong et al. [40] leveraged reinforcement learning for path-finding in knowledge graphs, while Das et al. [7] proposed MINERVA which makes use of a KG and trains a model for question answering. Lin et al. [20] introduced RL-based models for KG question answering with reward shaping. These approaches formulate multi-hop reasoning as a sequential decision-making problem, taking advantage of the structure and information present in knowledge graphs.

However, to the best of our knowledge, there has been limited research on utilizing RL in knowledge graphs specifically for the task of recommendation, especially when considering the challenge of navigating an extremely large action space as the number of path hops grows. This opens up a promising research direction for developing RL-based recommender systems that can exploit the rich information present in knowledge graphs while efficiently navigating large action spaces to provide personalized and explainable recommendations.

Reinforcement learning presents a promising avenue for recommender systems, particularly when combined with the rich information present in knowledge graphs. By exploring novel techniques for managing large action spaces, incorporating graph neural networks, and leveraging transfer learning, researchers can continue to push the boundaries of RL-based recommender systems, providing increasingly accurate, personalized, and explainable recommendations.

3 METHODOLOGY

The problem addressed in this research is to provide a new type of recommendation, called Knowledge Graph Driven Explainable Recommendation (KGDExR), that simultaneously performs item recommendation and path finding based on rich and heterogeneous information in the knowledge graph.

The goal is to find a recommendation set of N items for a given user u from a subset of Item entities $\mathbf{I}$ connected to User entities $\mathbf{U}$ through relations r_{ui} in The knowledge graph $\mathbf{G}$. The recommendation set should be associated with one reasoning path $p_j(u, i_n)$ $(2 \leq j \leq J)$ for each pair (u, i_n) of user and recommended item, where j is the number of hops in the path and J is a given integer. The number of recommendations, N, is also given as an input. The knowledge graph $\mathbf{G}$ is defined as $\mathbf{G} = (e^h, r, e^t)$, where e^h is the head entity and e^t is the tail entity in the KG. e^h & $e^t \in \mathbf{E}, r \in \mathbf{R}$, where $\mathbf{E}$ is the entity set and $\mathbf{R}$ is the relation set. A j-hop path from entity e_0 to entity e_j is defined as a sequence of $j+1$ entities connected by j relations, denoted by $p_j(e_0, e_j) = \left\{e_0 \overset{r_1}{\leftrightarrow} e_1 \overset{r_2}{\leftrightarrow} \dots \overset{r_j}{\leftrightarrow} e_j\right\}$.

The KGDExR problem can be formalized as finding a set of N items $\{i_n\}_{n \in [\mathrm{N}]} \subseteq \mathbf{I}$ for a given user u and integers J and N, such that each pair (u, i_n) is associated with a reasoning path $p_j(u, i_n)$ $(2 \leq j \leq J)$.

3.1 KG-Driven Reinforcement Learning based Recommender System

We use Markov Decision Process (MDP) framework to address the KGDExR problem. To ensure path connectivity, we supplement the graph $\mathbf{G}$ with two distinct types of edges. Primarily, reverse edges are included, such that if $(e^h, r, e^t) \in \mathbf{G}$, then $(e^t, r, e^h) \in \mathbf{G}$, aiding in the path definition.

The state at a given step t, denoted as s_t, is represented as a triplet (e_u, e_{s_t}, h_t), where $e_u \in U$ denotes the initial user entity, e_{s_t} indicates the entity the agent has arrived at step t, and h_t refers to the history before step t. We define the k-step history as the combination of all entities and relations in the previous k steps, i.e., $\left\{e_u \overset{r_j}{\leftrightarrow} e_j \overset{r_{j+1}}{\leftrightarrow} \dots \overset{r_{j+k-1}}{\leftrightarrow} e_{k-1} \overset{r_{j+k}}{\leftrightarrow} e_k\right\}$. Given some user u, the initial state is represented as $s_0 = (e_u, e_u, \emptyset)$ and the terminal state is represented as $s_T = (e_u, e_T, h_T)$.

The action space A_t at state s_t is defined as all possible emerging edges from an entity et. Some nodes in the KG can have very large out-degree which can make it inefficient to maintain the large action space. Therefore, we perform an action-pruning step based on a scoring function $f((r, e)|u)$, which maps any relation to a real-valued score conditioned on a given user [39]. There is a user-defined integer α that upper bounds the size of the action space. For our experiments, we set the value of $\alpha = 3$.

For a given user, a simple binary reward function is not appropriate as we don't know whether the agent has reached a target item or not. Therefore, the agent needs to find as many reasoning paths as possible. We consider giving a reward to the last state (s_T) of the path. The reward R_T is defined as:

$$R_T = \begin{cases} max\left(0, \frac{f(u, e_T)}{max_{i \in I} f(u, i)}\right), & \text{if } e_T \in I, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

In accordance with the underlying properties of the graph, the state in our recommendation system is determined by the entity's position. Given a state $s_t = (e_u, e_t, h_t)$ and an action $a_t = (r_{t+1}, e_{t+1})$, the transition to the next state s_{t+1} is characterized by a probability distribution:

$$P[s_{t+1} = (e_u, e_{t+1}, h_{t+1}) | s_t = (e_u, e_t, h_t), a_t = (r_{t+1}, et + 1)] = 1 \tag{2}$$

However, there is an exceptional case for the initial state $s_0 = (e_u, e_u, 0)$, which introduces stochasticity and depends on the starting user entity. To simplify the model, we assume a uniform distribution for the users, ensuring that each user is equally sampled at the beginning.

Building upon our Markov Decision Process (MDP) formulation, our primary objective is to learn a stochastic policy π that maximizes the expected cumulative reward.

We define the expected cumulative rewards based on all the paths traversed by a user as below:

$$J(\theta) = \mathbb{E}_{e_0} \in u[\mathbb{E}_{a_1, a_2, \dots, a_T \sim \pi_\theta(a_t|s_t)}[R_T]] \tag{3}$$

To maximize the expected cumulative rewards, we use gradient ascent. The gradients are derived by the REINFORCE [31], i.e.,

$$\nabla_\theta J(\theta) \approx \nabla_\theta \sum_t R_T log \pi_\theta(a_t|s_t). \tag{4}$$

The final step of our recommendation problem solution involves using a trained policy network to guide the exploration of a knowledge graph. Our objective is to find a set of candidate items and their corresponding reasoning paths for a given user. One approach is to sample paths for each user based on the policy network's guidance. However, this method may lack path diversity because the agent tends to repeatedly search the path with the highest cumulative rewards. To address this, we propose using Path Directed Reasoning (PDR) algorithm, which considers both action probability and reward, to explore candidate paths and recommended items for each user. The process is outlined in Algorithm 1. The algorithm takes inputs such as the KG, the user, and the policy network. The output is a set of T-hop paths for the user, along with their generative probabilities and rewards. Each path ends with an item entity and associated generative probability and reward. Among the candidate paths, there may be multiple paths between the user and an item. To interpret the reasoning behind why an item is recommended to the user, we select the path from the candidate set with the highest generative probability based on the generative probabilities. Finally, we rank the selected interpretable paths based on their path rewards and recommend the corresponding items to the user.

Algorithm 1: Path Directed Reasoning (PDR) Algorithm

```
Data: KG $G$, items $I$, users $U$; policy $\pi$
Result: Reward $R$ ; path $P$; probability $X$
Initialize $R$, $P$ and $X$;
for all $u \in U$ do
    for $t = 1$ to $T$ do
        Initialize $R_t = \phi, P_t = \phi, X_t = \phi$ ;
        for $\hat{p} \in P, \hat{r} \in R, \hat{x} \in X$ do
            Path $\hat{p} = \{e_u, r_1, ..., r_{t-1}, e_{t-1}\}$;
            Set state $s_{t-1} = (e_u, e_{t-1}, h_{t-1})$;
            Get pruned action space $\hat{A}_{t-1}(u)$;
            Get a path for action $a$ such that
              $p(a) = \pi(a|s_{t-1})$;
            Actions $A_t = \{a|rank(p(a))\}$;
            for all $a \in A_t$ do
                Get state $s_t$ and $R_t$;
                Assign new path $\hat{p} \cup \{r_t, e_t\}$ to $P_t$;
                Assign new probability $p(a)\hat{x}$ to $X_t$;
                Assign new reward $R_{t+1} + \hat{r}$ to $R_t$;
            end
        end
    end
    Export all paths that end with an item $i \in I$;
    Return updated $P_T, X_T$ and $R_T$ ;
end
```

3.2 KG-Driven XGBoost based Recommender System

XGBoost (eXtreme Gradient Boosting) [6] is an ensemble learning algorithm that has become a popular and effective method for a wide range of machine learning tasks, including classification, regression, and ranking. XGBoost builds a set of decision trees iteratively, using a gradient boosting approach to minimize a user-specified loss function.

For a dataset $D = \{(\mathbf{x}_i, y_i)\} \,|(\mathbf{x}_i \in \mathbb{R}^m, y_i \in \mathbb{R})$ that has n observations and m features, the XGBoost model uses Z additive functions for prediction [6].

$$\hat{y}_i = \sum_{z=1}^{Z} f_k(\mathbf{x}_i), \tag{5}$$

where $f_k \in F$ and F is the space of regression trees which can be defined as:

$$F = \left\{f(\mathbf{x}) = w_{q(x)}\right\} (q : \mathbb{R}^m \rightarrow T, w \in \mathbb{R}^T), \tag{6}$$

where q is the structure of each tree that maps an observation to the corresponding leaf node in the tree, T represents the number of leaf nodes in the tree, and w represents the leaf weights. For a given observation, the final prediction is computed by taking the sum of all the weights for the corresponding leave nodes.

The key idea behind XGBoost is to iteratively add decision trees to the ensemble, with each new tree trained to correct the residual errors of the previous trees. In other words, XGBoost fits the model by adding new trees to the ensemble that improve the overall prediction accuracy, while penalizing trees that are too complex or overfit the data.

One of the important features of XGBoost is its support for a wide range of objective functions and evaluation metrics, including common loss functions like squared error and logistic loss, as well as custom loss functions. XGBoost also includes a variety of regularization techniques to prevent overfitting and improve generalization performance, including L1 and L2 regularization terms, tree depth constraints, and early stopping.

For our initial experiments, we implemented three rankers within the XGBoost model to predict the ranking of the articles for the users. These are XGBoost ranker [6], CatBoost ranker [26], and LightGBM ranker [16]. CatBoost [26] is a recent library known for its efficacy in handling categorical features, which employs YetiRank [11] as the loss function. LightGBM [16] handles categorical features and optimizes the LambdaRank loss. We trained XGBoost ranker [6] with Rank Pairwise loss, utilizing one-hot encoding. During our initial experiments, the XGBoost ranker outperformed the other two rankers. Therefore, we selected the XGBoost ranker for our KG-driven XGBoost-based recommender system approach.

We used XGBoost ranker in combination with KGs generated from article text and the other article features to build the XGBoost-based recommender system. The KGs generated are then used as input to the TuckER and TransE to generate 300-dimensional KG embeddings. These embeddings along with the customer demographical data and article features are used to train the KG-driven XGBoost-based recommender system.

4 EXPERIMENTAL SETUP

In this section, we provide information on KG creation, KG embedding generation, and the data sets used in this work.

4.1 Automatic KG Generation

To automatically generate KGs from the targeted unstructured data sets, we used two approaches. The first approach makes use of external lexical resources, such as ConceptNet [30] to connect terms and enrich the taxonomy. The second approach is different in the way that it neither requires any training nor any external resource, but instead uses the knowledge of the domain available within the input data to extract the knowledge.

4.1.1 *ConceptNet-based approach.*

ConceptNet [30] is a knowledge graph that encompasses entities from various domains along with their corresponding relationships. For this study, we specifically focus on three relationship types: IsA, PartOf, and Synonym. The "IsA" relationship signifies hypernymy relations, while "PartOf" represents meronymy relations, and "Synonym" indicates synonymy relations. To generate a dataset for hyponymy relations, we inverted the direction of relations labeled as hypernyms. All other relations in ConceptNet were grouped together as "other." The training dataset was created by including all extracted relationships.

The system architecture is based on BERT [8], employing 12 transformer blocks. The embeddings utilized are extracted from the transformer in the 12th layer. Pretrained embeddings from the BERT model "uncased_L-24_H-1024_A-16" are employed, which are readily available in TensorFlow. We named "uKG_CN" to the KG that we generated using the ConceptNet-based approach.

4.1.2 *Dependency Parsing-based approach.*

The creation of a domain-specific KG with this approach follows a mixed approach based on both the Saffron tool[2] for taxonomy generation and the new algorithm for relation extraction. It uses the syntactic knowledge of sentences in a textual dataset to extract new relations between Saffron terms. After extracting the new relations from the text, we integrate them into the Saffron taxonomy and return a fully formed KG. This approach does not require any training and is domain independent.

The dependency parsing-based relation extraction approach extracts relations from the text and exports them as triples (left_relation, relation_type, right_relation). It uses dependency parsing (syntactic analysis of the sentences) on the text to find how terms are syntactically (and by extension semantically) connected within sentences. It takes as input the terms extracted by Saffron [25], as well as the dataset originally used to extract the Saffron terms and extract the taxonomy, and returns a list of triples: term1, relation, term2. The whole implementation is done in Python. We named "uKG_DP" to the KG that we generated using the Dependency Parsing based approach.

We have also created a KG, referred to as "uKG", from unstructured data. This KG contains only the article and its relation with the most frequent terms found within the article. To compute the Term Frequency, we utilized TF-IDF.

4.1.3 *KG creation using both structured and unstructured data (cKG).*

We have already defined the (KGDExR) problem and provided the definition of a KG in section 3. Here, we will illustrate how we constructed KG using both structured and unstructured data (combined data (cKG)). The features of structured data, such as 'user', 'article', 'topic', 'product', 'topic_tag', 'product_tag', 'response', etc., serve as the type of nodes or entities in the KG. These entities are connected to other entities through relations such as 'has_topic', 'has_product', 'has_topic_tag', 'has_product_tag', and 'has_response'. Additionally, we utilized the full text of the article, which represents the unstructured data, to create this KG. Therefore, this KG leverages both structured and unstructured data for its creation. The recommendation process begins with a user, traverses through specific entities and their associated relations, and ultimately leads to an item, which in our case is the recommended article for that user. We have named the KG generated using structured and unstructured data that is the combined data as "cKG".

4.2 Knowledge Graph Embeddings

In a given KG, each head entity or tail entity can be associated as a point in a continuous vector space. In this work, we use TuckER [2, 3] and TransE [38] methods to generate KG embeddings. TuckER employs a three-way TuckER tensor decomposition, which computes the tensor T and a sequence of three matrices leveraging the embeddings of entities (E_{head} and E_{tail}) and relations (R) between them ($G \approx T \otimes E_{head} \otimes R \otimes E_{tail}$).

The underlying idea of TransE is to interpret relations as translations that occur between entities in the knowledge graph. In TransE, each entity and relation is assigned a unique vector representation in the embedding space. The objective of the model is to learn these embeddings in such a way that the translation between the embeddings of a head entity and a relation should be close to the embedding of a tail entity. These methods allow us to create KG embeddings that are used to train our recommender systems.

4.3 Data sets

The dataset used in this study contains the data of the customers of a large multinational financial services company and the viewpoint articles sent to these customers by the company. The dataset spans from January 30th, 2019 to October 30th, 2019, and contains information of 463 customers who received approximately 80 articles each during this period. The dataset consists of 37,423 rows, detailing individual customer-article interactions. It includes a total of 71 articles, with 66 unique articles, providing details related to the products and services that the financial services company provides. This dataset serves as a valuable resource for researchers and marketers interested in understanding customer behavior and preferences, as well as identifying opportunities for targeted content and marketing strategies. We used this dataset for the evaluation of our KG-driven RL-based approach and KG-driven XGBoost approach for recommending articles to customers. The dataset is divided into training, and test sets with a ratio of 70:30 respectively. We have also made this data set publicly available on a GitHub repository [3].

5 RESULTS

We have produced results using both KG-driven XGboost approach and KG-driven reinforcement learning approach.

Table 1 represents the results obtained using the proposed approaches with the KG embeddings used for the model building. From Table 1, we can see that the baseline XGBoost model with sentence transformer embedding [all-MiniLM-L6-v2] achieved a 30.38% MAP score. We observed improvements in performance when we used KG embeddings compared to when KG embeddings were not used (see Tabel 1).

We constructed two KGs using unstructured data (article text) through Saffron [25] as mentioned in Section 4. These KGs are

[2]https://saffron.insight-centre.org/

[3]https://github.com/GhanshyamVerma/Explainable-Recommender-System.

Table 1: Results of KG-driven XGBoost based Recommender system and KG-driven RL based Recommender system with baseline XGBoost approach.

Model	Embedding	MAP@K=10	Precision@K=10	Recall@K=10
BPR (Bayesian personalized ranking)	-	0.11207	0.05672	0.41904
Neighborhood-based Recommender System	-	0.20418	0.66177	0.27175
NCF (Neural Collaborative Filtering)	-	0.24104	0.62599	0.30007
XGBoost	Sentence Transformer Embedding [all-MiniLM-L6-v2]	0.30381	0.65902	0.21568
KG-XGBoost [uKG_DP]	Saffron Dependency Parsing KG (TuckER KG Embedding)	0.34378	0.71708	0.23965
KG-XGBoost [uKG_CN]	Saffron ConceptNet KG (TuckER KG Embedding)	0.38985	0.24575	0.74384
KG-XGBoost [uKG]	TransE KG Embedding	0.33774	0.23137	0.70987
KG-XGBoost [cKG]	TransE KG Embedding	0.34468	0.24031	0.73740
KG-RL [cKG]	TransE KG Embedding	0.43761	0.60562	0.24857

"uKG_DP" and "uKG_CN" where u denotes unstructured data, DP denotes dependency parsing and CN denotes ConceptNet. Additionally, we created a KG referred to as "cKG" from both structured and unstructured data, as explained in Section 4.

The rationale behind using the cKG with RL-based approach is that it helps in generating explainable recommendations using paths in the cKG. For RL based approach we used KG embeddings generated using TransE, as shown in Table 1.

We also compared the performance of our proposed approaches with state-of-the-art existing recommender systems. The existing recommender systems we used are: BPR (Bayesian personalized ranking), Neighborhood-based Recommender System, NCF (Neural Collaborative Filtering), and XGBoost with sentence embedding. We observed that BPR achieved a MAP score of 11.21%, whereas the KG-driven XGBoost approach (cKG) and KG-driven RL-based approach using the same cKG achieved 34.47% and 43.76% MAP scores, respectively. The KG-driven XGBoost approach with KG generated using ConceptNet achieved a MAP score of 38.98% with a recall of 74.38%. The results suggest that if recall is important for any application, then KG-driven XGBoost with uKG_CN can be considered as an option, as it provides the highest recall. Based on the results, it can be observed that the KG-driven RL-based approach outperformed the BPR, Neighborhood-based Recommender System, NCF, and KG-driven XGBoost approaches when considering the MAP score. Additionally, among all the experiments conducted with KG embeddings, the KG embeddings generated from TransE have proven to capture useful information, resulting in better performance compared to TuckER embeddings.

Our KG-driven RL-based approach is explainable. To gain a better understanding of our model's interpretation of the recommendation, we present a case study based on the results obtained from our experiments. In this study, we analyze the path patterns uncovered by our model during the reasoning process, as well as

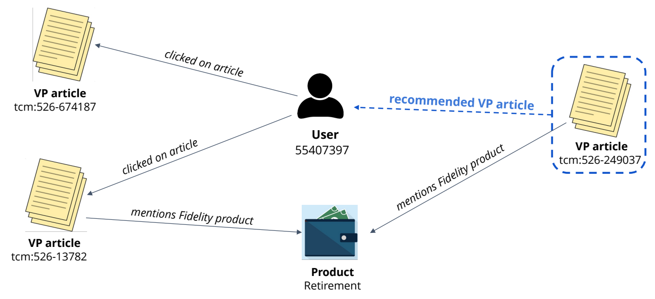


Figure 1: Explaining the recommendations of RL-based approach using the path in the KG that leads to the recommendation.

examine different recommendation scenarios. As shown in Figure 1, the article highlighted with a blue dashed boundary is the article recommended by our RL-based model to a user. We can see that the recommended article has some similarities with another article already recommended and clicked by that user, therefore the model thinks that this article should be of relevance for that user as the user was interested in such kind of articles before. Furthermore, our RL-based approach enables us to offer the top 10 articles for each user. Additionally, it can provide all the associated articles in the path that lead to the outcome, along with shared products, topics, and the most frequent common terms found in the text of the articles present in the path. Our RL-based approach can provide such paths for each recommended item to a user which explain the results and play an important role in decision-making.

To generate post-hoc explanation for KG driven XGBoost-based approach, we used SHAP [22] and ELI5[4]. SHAP (SHapley Additive

[4]https://github.com/TeamHG-Memex/eli5

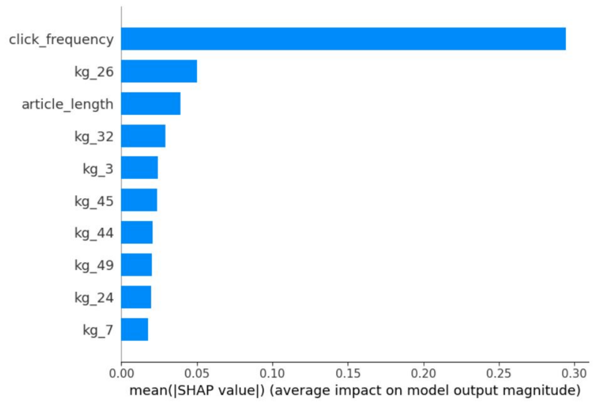
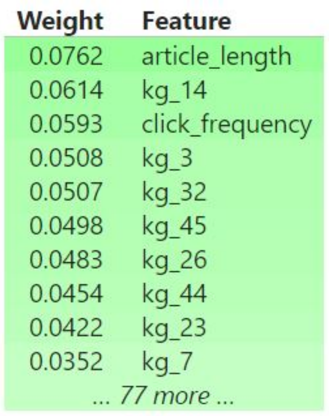


Figure 2: Explaining the recommendations of KG-XGBoost [uKG_CN] model using SHAP.



Weight	Feature
0.0762	article_length
0.0614	kg_14
0.0593	click_frequency
0.0508	kg_3
0.0507	kg_32
0.0498	kg_45
0.0483	kg_26
0.0454	kg_44
0.0422	kg_23
0.0352	kg_7
	... 77 more ...

Figure 3: Explaining the recommendations of KG-XGBoost [uKG_CN] model using ELI5.

exPlanations) is a model-agnsotic method used for explaining the output of machine learning models. It is based on game theoretic concepts and provides an explanation for each feature's contribution to the model's prediction. SHAP values quantify the impact of each feature by assigning a value to it, indicating how much it contributes to the prediction compared to the average prediction. SHAP relies on the concept of Shapley values from cooperative game theory and it considers additive feature importance. Figure 2 represents the KG-XGBoost [uKG_CN] model's features with their average impact on the model output generated by SHAP.

ELI5 (Explain Like I'm 5) is a Python library or framework for explainable machine learning models. ELI5 focuses on understanding the overall behavior and importance of features in making predictions. Eli5 reports feature importance using the "permutation importance" algorithm. Figure 3 shows the KG-XGBoost [uKG_CN] model's feature importance by assigning weights to the features based on their impact on the model output generated by ELI5. Both SHAP and ELI5 show that click_frequency, kg_26, article_length, kg_32, Kg_3, and Kg_45 are the most important features that contributed most to the model results.

Overall, the proposed approaches are helpful in providing insights to understand the recommendations and simultaneously perform better than the existing baseline recommender systems.

6 CONCLUSION

This research paper explores the use of knowledge graphs (KGs) to enhance personalized recommendations in the financial sector. We developed two KG-driven recommender systems for a large multinational financial services company, utilizing reinforcement learning and the XGBoost algorithm, respectively. The first approach employs Reinforcement Learning (RL), while the second utilizes the XGBoost algorithm. The XGBoost-based approach uses KG embeddings generated from both TuckER and TransE, and the RL-based approach uses TransE-generated embeddings. We also performed experiments keeping the KG and the embedding same. The findings suggest that the KG-driven RL-based approach outperforms both the KG-driven XGBoost system and baseline models, delivering more accurate and personalized article recommendations. Additionally, the study emphasizes the importance of reasoning with knowledge for decision-making. Overall, this study highlights the potential of combining advanced machine learning techniques with KG-driven insights to improve customer experience and drive business growth in the investment sector.

ACKNOWLEDGMENTS

This publication has emanated from research supported in part by a grant from Science Foundation Ireland under Grant number SFI/12/RC/2289_P2. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

REFERENCES

[1] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. *Algorithms* 11, 9 (2018). https://doi.org/10.3390/a11090137

[2] Mihael Arcan, Sampritha Manjunath, Cécile Robin, Ghanshyam Verma, Devishree Pillai, Simon Sarkar, Sourav Dutta, Haytham Assem, John P. McCrae, and Paul Buitelaar. 2023. Intent Classification by the Use of Automatically Generated Knowledge Graphs. *Information* 14, 5 (2023). https://doi.org/10.3390/info14050288

[3] Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. TuckER: Tensor Factorization for Knowledge Graph Completion. In *Empirical Methods in Natural Language Processing*.

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems* 26 (2013).

[5] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*. 151–161.

[6] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[7] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851* (2017).

[8] 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[9] Li Gao, Hong Yang, Jia Wu, Chuan Zhou, Weixue Lu, and Yue Hu. 2018. Recommendation with multi-source heterogeneous information. In *IJCAI International Joint Conference on Artificial Intelligence*.

[10] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[11] Andrey Gulin, Igor Kuralenok, and Dimitry Pavlov. 2011. Winning the transfer learning track of yahoo!'s learning to rank challenge with yetirank. In *Proceedings of the Learning to Rank Challenge*. PMLR, 63–76.

[12] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering* 34, 8 (2020), 3549–3568.

[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[14] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. 1999. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. 230–237.

[15] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 505–514.

[16] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017).

[17] Joseph A Konstan, Bradley N Miller, David Maltz, Jonathan L Herlocker, Lee R Gordon, and John Riedl. 1997. Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM* 40, 3 (1997), 77–87.

[18] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[19] Daniel Lee and H Sebastian Seung. 2000. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems* 13 (2000).

[20] Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-Hop Knowledge Graph Reasoning with Reward Shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). Association for Computational Linguistics, 3243–3253. https://doi.org/10.18653/v1/d18-1362

[21] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003), 76–80.

[22] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 4765–4774. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[23] Andriy Mnih and Russ R Salakhutdinov. 2007. Probabilistic matrix factorization. *Advances in neural information processing systems* 20 (2007).

[24] Maximilian Nickel, Volker Tresp, Hans-Peter Kriegel, et al. 2011. A three-way model for collective learning on multi-relational data.. In *Icml*, Vol. 11. 3104482–3104584.

[25] Bianca Pereira, Cecile Robin, Tobias Daudert, John P. McCrae, Pranab Mohanty, and Paul Buitelaar. 2019. Taxonomy Extraction for Customer Service Knowledge Base Construction. In *Semantic Systems. The Power of AI and Knowledge Graphs*, Maribel Acosta, Philippe Cudré-Mauroux, Maria Maleshkova, Tassilo Pellegrini, Harald Sack, and York Sure-Vetter (Eds.). Springer International Publishing, Cham, 175–190.

[26] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: unbiased boosting with categorical features. *Advances in neural information processing systems* 31 (2018).

[27] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. 175–186.

[28] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.

[29] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*. Springer, 593–607.

[30] Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 31.

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

[32] Georgios Theocharous, Philip S Thomas, and Mohammad Ghavamzadeh. 2015. Ad recommendation systems for life-time value optimization. In *Proceedings of the 24th international conference on world wide web*. 1305–1310.

[33] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 1225–1234.

[34] 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*. 417–426.

[35] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. 2018. A reinforcement learning framework for explainable recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 587–596.

[36] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[37] 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.

[38] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 28.

[39] Yikun Xian, Zuohui Fu, Shan 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.

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

[41] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 353–362.

[42] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W Bruce Croft. 2017. Joint representation learning for top-n recommendation with heterogeneous information sources. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1449–1458.

[43] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 world wide web conference*. 167–176.