Improving Transformer-based Sequential Conversational Recommendations through Knowledge Graph Embeddings

Alessandro Petruzzelli*
alessandro.petruzzelli@uniba.it
University of Bari
Bari, Italy

Alessandro Francesco Maria Martina*
a.martina13@studenti.uniba.it
University of Bari
Bari, Italy

Giuseppe Spillo
giuseppe.spillo@uniba.it
University of Bari
Bari, Italy

Cataldo Musto
cataldo.musto@uniba.it
University of Bari
Bari, Italy

Marco de Gemmis
marco.degemmis@uniba.it
University of Bari
Bari, Italy

Pasquale Lops
pasquale.lops@uniba.it
University of Bari
Bari, Italy

Giovanni Semeraro
giovanni.semeraro@uniba.it
University of Bari
Bari, Italy

ABSTRACT

Conversational Recommender Systems (CRS) have recently drawn attention due to their capacity of delivering personalized recommendations through multi-turn natural language interactions. In this paper, we fit into this research line and we introduce a Knowledge-Aware Sequential Conversational Recommender System (KASCRS) that exploits *transformers* and *knowledge graph embeddings* to provide users with recommendations in a conversational setting.

In particular, KASCRS is able to predict a suitable recommendation based on the elements that are mentioned in a conversation between a user and a CRS. To do this, we design a model that: *(i)* encodes each conversation as a sequence of entities that are mentioned in the dialogue (*i.e.*, items and properties), and *(ii)* is trained on a *cloze* task, that is to say, it learns to predict the final element in the sequence - that corresponds to the item to be recommended - based on the information it has previously seen.

The model has two main hallmarks: first, we exploit Transformers and *self-attention* to capture the sequential dependencies that exist among the entities that are mentioned in the training dialogues, in a way similar to session-based recommender systems [25]. Next, we used *knowledge graphs* (KG) to improve the quality of the representation of the elements mentioned in each sequence. Indeed, we exploit knowledge graph embeddings techniques to pre-train the representation of items and properties, and we fed the input layer of our architecture with the resulting embeddings. In this way, KASCRS integrates both knowledge from the KGs as well as the dependencies and the co-occurrences emerging from conversational data, resulting in a more accurate representation of users and items. Our experiments confirmed this intuition, since KASCRS overcame several state-of-the-art baselines on two different datasets.

*Both authors contributed equally to this research.

CCS CONCEPTS

• **Information systems** → *Users and interactive retrieval*; **Recommender systems**; *Task models.*

KEYWORDS

Conversational Recommendations, Transformers, Knowledge Graphs, Recommender Systems

ACM Reference Format:
Alessandro Petruzzelli, Alessandro Francesco Maria Martina, Giuseppe Spillo, Cataldo Musto, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. 2024. Improving Transformer-based Sequential Conversational Recommendations through Knowledge Graph Embeddings. In *Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization (UMAP '24), July 01–04, 2024, Cagliari, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3627043.3659565

1 INTRODUCTION

Conversational Recommender Systems (CRS) are intelligent systems designed to support decision-making [11] by interacting with the users in a multi-turn dialogue [21], and by providing personalized recommendations based on their preferences and feedback [17]. In recent years, CRSs have gained significant attention, mainly due to the progress in the Natural Language Processing (NLP) field and to their potential in enhancing user satisfaction, trust, and engagement [16]. While early approaches for CRSs relied on manually curated knowledge [6–8, 20, 27, 35] and provided users with a *static* dialogue based on a relatively fixed sequence of questions, recent attempts focused on developing *end-to-end* approaches that

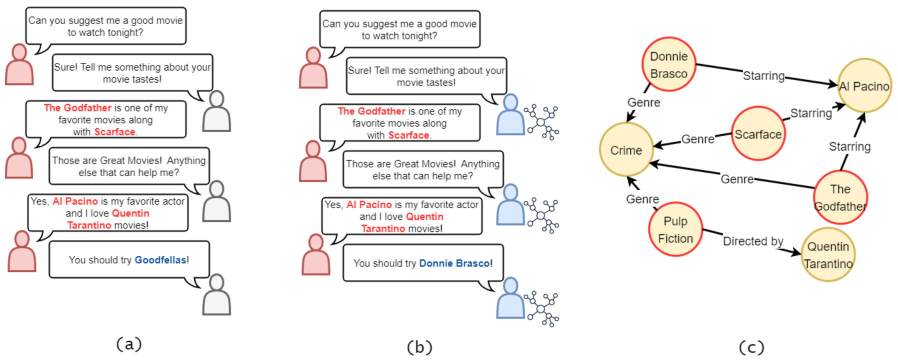

Can you suggest me a good movie to watch tonight?
Sure! Tell me something about your movie tastes!
The Godfather is one of my favorite movies along with Scarface.
Those are Great Movies! Anything else that can help me?
Yes, Al Pacino is my favorite actor and I love Quentin Tarantino movies!
You should try Goodfellas!
(a)
Can you suggest me a good movie to watch tonight?
Sure! Tell me something about your movie tastes!
The Godfather is one of my favorite movies along with Scarface.
Those are Great Movies! Anything else that can help me?
Yes, Al Pacino is my favorite actor and I love Quentin Tarantino movies!
You should try Donnie Brasco!
(b)
Donnie Brasco
Starring
Al Pacino
Genre
Starring
Scarface
Crime
Genre
Starring
Genre
The Godfather
Genre
Pulp Fiction
Directed by
Quentin Tarantino
(c)

Figure 1: Two different *toy* conversations run by a CRS. In the first case, the recommendation only relies on dependencies and co-occurrences learned from training dialogues. In the latter, the information coming from a KG (see *(c)* as a toy example) is also exploited.

are trained on datasets of previously annotated dialogues, such as ReDial [24]. Generally speaking, most of the end-to-end CRSs are composed by two main modules: a *conversation* module, whose goal is to maintain the dialogue with the user, by making the right questions and generating appropriate responses, and a *recommendation* module, whose goal is to collect and exploit all the information that is exchanged in the dialogue in order to provide users with a suggestion. In this paper, we focus on the *recommendation module* of a CRS, and we address the problem of extracting and encoding preferences and needs based on the sequence of natural language utterances that a user and a CRS exchange in a dialogue (see Figure 1).

The first attempt in this direction was proposed in [24], where the authors addressed this challenge by modeling each user based on the items explicitly mentioned in the dialogue. Later, the research extended this approach by also incorporating contextual words [29] and external knowledge from knowledge graphs (KG) [4, 28], as in [49]. This resulted in better recommendation performance. Another approach to user profiling, introduced in [9], modeled each user by exploiting items (*i.e.,* movies) and entities (*i.e.,* directors and genres) mentioned in the dialogue while employing a KG to build a connection between them. While being relatively accurate, all these methods share the common issue of treating the elements mentioned in the dialogue as a *set*, by overlooking the *sequential dependencies* that exist among the entities that appear in the dialogue (*i.e.,* the co-occurrences and the order in which they were expressed).

In order to better model such dependencies, methods such as the transformer-based sequential conversational recommender (TSCR) have been introduced [51]. TSCR established a new benchmark for CRS performance by modeling *sequential dependencies* among entities that appear in the dialogues. This model encodes the user profile as a sequence of entities, as in [9], but it also captures the positional information through *positional encoding*. While the use of positional embeddings currently represents a promising strategy to encode user preferences starting from a dialogue between a user and a CRSs, methods such as TSCR learn what to recommend by just exploiting the sequential dependencies and co-occurrences *that appear in the training dialogues*. In other therms, these models are trained on sequences of entities and their learning is solely dependent on the sequential patterns and co-occurrences of entities observed during training. As a consequence, they lack of the ability to generate recommendations by also relying on exogenous information coming from external *knowledge sources*, that can be useful to provide users with more diverse and accurate suggestions.

As an example, in a movie CRS scenario (see Figure 1-a), if a user has expressed a preference for *The Godfather* and for *Al Pacino* in their dialogue turns, other movies that appear in the training dialogues together with both the entities (*i.e.,* movies recommended to users who liked *The Godfather* and/or *Al Pacino*) will be candidate recommendations. Indeed, in our toy example, we assume that *Goodfellas* typically co-occurs in the sequences together with both *The Godfather* and *Al Pacino*, so it is returned as a recommendation. Unfortunately, such an approach does not exploit the information embedded in external structured knowledge sources such as KGs, which could enhance the accuracy of the model by exploiting the semantic relationships between the entities. Indeed, a CRS that also exploits a KG (see Figure 1-b) would return a different recommendation. In this case, by levering the information encoded in a KG (see Figure 1-c), a different movie, i.e., *Donnie Brasco*, could be recommended to the same user since it shares some semantic relationships with the preferences of the user, such as having the same genre of *The Godfather* (*Crime*) and being acted by the same actors the user likes (i.e., *Al Pacino*). In a nutshell, thanks to the exogenous information encoded in a KGs it is possible to provide CRSs with more *knowledge* that can be useful to generate more accurate recommendations, especially when training data are scarce and just a few co-occurrences may be exploited.

To address this limitation and to fully exploit the potential of KGs in CRS, we propose a Knowledge-Aware Sequential Conversational Recommender System (KASCRS), which leverages Knowledge Graph Embeddings (KGE) to pre-train the representation of items and properties in a CRS. To this end, we first learn a KGE for all the entities (*i.e.,* items and properties) in the dataset. Next, these pre-trained embeddings are fed into a deep architecture based on transformers that predicts a suitable recommendation. This allows KASCRS to not only capture sequential dependencies between entities via self-attention, but also to incorporate the semantic relationships between them based on the information available in the KG. As demonstrated by experimental evaluation, the combination of positional embeddings and exogenous knowledge based on pre-trained KGE significantly improves the predictive accuracy of a transformer-based sequential CRS, whose performance overcame those of several state-of-the-art baselines for conversational recommendations in two different experimental settings. As a final remark, as we previously stated, it is important to point out again that this work only focuses on the *recommendation* module of a CRS. This is line with other state-of-the-art work in the area, such as TSCR [51]. The design and the development of the *conversation* module of the architecture is left as a future work. To sum up, the contributions of this paper are the following:

(1) We introduce KASCRS, an approach to incorporate information encoded in KGs into a transformer-based sequential CRS, and we compared our approach to several state-of-the-art methods for CRSs.
(2) We carried out a sensitivity analysis by comparing the performance of the approach on varying of the graph embedding technique and on varying of different hyper-parameters;
(3) We guarantee reproducibility by releasing our source code and all the scripts to run the experiments.

The remainder of this paper is organized as follows: Section 2 provides an overview of the state of the art in CRS and KGE; Section 3 details the data processing steps and outlines the problem we aim to address; Section 4 focuses on the knowledge graph embedding and Section 5 describes the architecture of of the proposed knowledge-aware sequential CRS. Finally, Section 6 presents the experimental results.

2 RELATED WORKS

Conversational Recommender Systems. CRSs have emerged as a popular research topic in recent years, aiming to provide high-quality recommendations to users through natural language conversations [24, 26]. In the literature, two primary categories of CRSs have been investigated: attribute-based CRSs [14] and end-to-end CRSs [24].

Generally speaking, *attribute-based CRSs* prioritize capturing user preferences and generating precise recommendations in a limited number of turns [10, 40, 50]. These systems interact with users through pre-defined actions and generate responses using templates. Recent works have employed multi-armed bandit [10] or reinforcement learning algorithms [23] to enhance user interactions. However, these models require a knowledge engineering step, which restricts the flexibility and the adaptivity of the proposed solutions [15, 22]. On the other side, *end-to-end CRSs* typically rely on large amount of training dialogues and exploit deep neural networks. The distinctive trait of these approaches lies in the quality of the dialogues and the natural language responses they can generate. Indeed, these systems are more challenging to develop but can provide a more engaging and personalized user experience [21].

Within the context of end-to-end CRSs based on neural networks, the issue of modeling user preferences has been addressed in different ways, depending on the architecture underpinning the recommendation module. As an example, some state-of-the-art-approaches exploited pre-trained language models (PLM) for this task. In particular, in [48] the authors introduce some implementations of CRSs based on GPT-2 [34] and based on BERT [12]. In these models, the user profile is obtained by feeding the PLM will *all* the tokens that appear in the conversation, and the model is trained to predict the final element (*i.e.,* the item to be recommended). In particular, as for GPT-2, the recommendations are generated by using the representation of the final token while representation of the "[CLS]" token is used for BERT. While being relatively simple, these approaches provide good predictive accuracy, so we considered these techniques as *baselines* in our experiments. Next, more advanced strategies to handle conversations in CRS recently emerged. In ReDial [24], the authors employ an auto-encoder recommendation module pre-trained on MovieLens [18]. This recommendation module provides suggestions based on the *items* mentioned in the dialogue only. Next, several approaches exploited KGs to generate recommendations in CRSs. As an example, in KGSF [49] the authors employ a KG-enhanced recommendation module that considers both items and contextual words mentioned in the dialogues. Moreover, this approach also exploits embeddings learned from DBpedia [2] and ConceptNet [37] to inject exogenous knowledge related to items and contextual words, respectively. Next, a different perspective is presented in KBRD [9]. In this work, each user is modeled using entities (items and properties) appearing in the dialogue. Next, items are linked to the corresponding URI in DBpedia KG, while properties (*i.e.,* actors and directors) are linked to the corresponding URI in the same KG. However, this method, along with prior approaches, fails to consider the sequential order in which user preferences are elicited, differing from our proposed method. To sum up, in all these approaches each user is thus represented using the set of entities mentioned in the dialogue, and this representation is enriched by structural and relational information learned from the KG. Recently, the use of positional information emerged as a promising strategy to encode user preferences. Indeed, as previously stated, transformer-based sequential conversational recommenders exploited this intuition and have shown state-of-the-art performance in this task. In TSCR [51], the authors propose a recommendation module inspired by BERT4Rec [39]. In this work, users are modeled by extracting entities, which encompass items and properties (*i.e.,* directors and genres), mentioned in natural language dialogues. Then, these entities are then modeled as a *sequence.* In this way, they explicitly consider order which entities are mentioned in the dialogue. As shown in [51], the use of positional embeddings to capture sequential dependencies between entities led TSCR to obtain the best results w.r.t. other baselines in the area of CRS.

To sum up, the overview of the literature showed that state-of-the-art approaches fit into two different research lines: on the one side, we have methods that exploit exogenous knowledge sources to enrich the representation of the entities mentioned in the dialogue. On the other, we have techniques such as TSCR that leverage the sequential nature of the dialogues and exploit this information to improve the predictive accuracy. While both the research lines obtained promising results, the combined exploitation of both the intuitions (*i.e.,* use of knowledge graphs and use of positional information) is poorly investigated. Accordingly, we introduce a knowledge-aware sequential conversational recommendation module that generates recommendations based on entities mentioned in the dialogue. Unlike previous methods, our model effectively captures both sequential dependencies and relational/structural information from the conversational dataset and KG, respectively, providing users with a more comprehensive and accurate recommendations.

Knowledge Graph Embeddings. Graph embedding techniques aim at representing *entities* and *relations* holding in a graph as *dense vectors* (or *embeddings*), while preserving properties and structures of the original graph. The first proposed models, such as TransE [5] and TransH [46], are referred to as *geometrical models* [44]. While being very simple, these models obtained very good performance in several tasks, including recommendation [31, 32, 38]. More recently, these models evolved by considering Large Language Models [3] and by exploiting Graph Neural Networks and Graph Convolutional Networks (GCNs) [47]. As for the latter, the key idea of these models is to learn node embeddings by propagating and aggregating neighbors-derived information. CompGCN [42] is a GCN model that performs a composition operation over each edge in the neighborhood of a target node, and then applies convolution on the composed embeddings. Based on the very competitive performance obtained by GCNs in recommendation tasks [33, 38], we exploited GCNs to learn our KGE. However, with respect to most of the literature in the area of KGE for recommender systems, the novelty of our work lies in the use of KGE to feed a transformer-based sequential CRS. As we previously stated, this is a poorly investigated research line.

3 PRELIMINARIES

Knowledge Graph. All the information necessary to learn our pre-trained representation of items and properties is encoded in a bipartite knowledge graph $\mathcal{KG}$. In this graph, nodes representing items and descriptive properties are connected by labeled edges, reflecting the links between them. An example of $\mathcal{KG}$ is provided in Figure 1-c.

Formally, let $\mathcal{I} = \{i_1, i_2, \ldots, i_{|\mathcal{I}|}\}$, be the set of items and $\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\}$ be the set of properties such as actors, directors and genres. $\mathcal{KG} = \langle \mathcal{N}, \mathcal{E} \rangle$, where $\mathcal{N} = \mathcal{I} \cup \mathcal{F}$. Next, let $\mathcal{R}$ be the set of the relations that exist in the graph (such as *starring*, *directed by*, etc.). If item $i \in \mathcal{I}$ is linked to property $f \in \mathcal{F}$ through relation $r \in \mathcal{R}$, then $(i, r, f) \in \mathcal{E}$, which means that a direct edge connecting i to f, and labeled with r, is created.

Conversational Data. Let $\mathcal{U} = \{u_1, u_2, \ldots, u_{|\mathcal{U}|}\}$ be a set of users. Given a dataset of dialogues $\mathcal{D}$ describing conversations between a user and a CRS, for each $d \in \mathcal{D}$ we extract the sequence of entities mentioned by the user in the dialogue. Formally, each dialogue is encoded as a sequence entity tokens $\mathcal{S}_{u,d} = [e^d_1, e^d_2, \ldots, e^d_k]$ where $e^d_i \in \{\mathcal{I} \cup \mathcal{F}\}$. $\mathcal{S}_{u,d}$ represents the preferences of user u extracted from d. The extraction phase can employ any entity recognition and entity linking techniques, and follows the setup used in [9, 49]. The whole process is described in Figure 2.

Description of the Problem. Given a knowledge graph $\mathcal{KG}$, we first employ KGE techniques to learn an embedding $V(e)$ for each entity $e \in \{\mathcal{I} \cup \mathcal{F}\}$. Next, let $\mathcal{S}_{u,d}$ be a sequence of entities representing the preferences of user u extracted from dialogue d. The objective of the model is to predict the probability that a user would like the item e_j.

Formally, the model learns a function $\mathcal{Z}$ such that $y_j = \mathcal{Z}(\mathcal{S}_{u,d}, e_j | V, \theta)$ where θ are the model's learnable parameters, V is a mapping function that links each entity to its corresponding KGE, and e_j is the candidate item. This prediction is calculated for all the candidate items in I, the scores y_j are ranked in descending order and the top-1 item is returned as conversational recommendation.

4 KNOWLEDGE GRAPH EMBEDDING LEARNING

Starting from the bipartite knowledge graph $\mathcal{KG}$, we first learn the graph embeddings for all the nodes in the KG. As previously stated, we learn KGE by using the Graph Convolutional Network [47] CompGCN [42]. The general idea of this model is to aggregate the information coming from the neighborhood of a specific node to learn its representation, and this is repeated for each node up to a specific l-th layer, where l is the number of layers of the GCN. The main advantage of this model, with respect to other GCNs, is that it is able to distinguish different relations and treat them in different ways, aiming at learning more precise embeddings. Formally, given a node e, we refer to its vector-space representation with $V(e)$; then, the embedding is updated as follows:

$$V(e)^{k+1} = \sum_{(f,r) \in N(e)} W^k_{\lambda(r)} \Phi\left(V(f)^k, V(r)^k\right) \tag{1}$$

where $V(e)^{k+1}$ is the embedding of the node e learned at the $k+1$-th layer, $N(e)$ is the set of neighbors e of the node f linked via the relation r, $W^k_{\lambda(r)}$ is the transformation matrix of the relation r, used to project the relation embedding to the same space of node embeddings and use them in the next layer (it also considers the direction of the edge by means of the function $\lambda(r)$), $V(f)^k$ and $V(r)^k$ are the embeddings related to the node f and the relation r learned at the previous layer, respectively, and finally $\Phi\left(V(f)^k, V(r)^k\right)$ is the composition function aiming at combining the embeddings. As in TransE [5], we used the subtraction as composition function. For more details on the method, we suggest to refer to the original paper [42]. While our choice is based on the very competitive performance obtained by GCNs in recommendation [33, 38], in our experimental setting we also compared the embeddings learnt with CompGCN with those obtained with other KGE strategies. Moreover, we also evaluated the effectiveness of the embeddings learnt at different layers. More details will be provided next.

5 KNOWLEDGE-AWARE SEQUENTIAL CONVERSATIONAL RECOMMENDER SYSTEM

The design and the implementation of our transformer-based sequential CRS follows the description presented in Section 3. In the following, we present our architecture and we describe training and inference of the model.

Symbol	Definition	Description
$\mathcal{KG}$	Knowledge Graph	Source of external knowledge that is leveraged to obtain pre-trained embeddings
$\mathcal{N}$	Nodes of $\mathcal{KG}$	Nodes are heterogeneous, including items and descriptive properties
$\mathcal{I}$	Set of items	In the movies domain, it includes only movies
$\mathcal{F}$	Set of descriptive properties	In the movies domain, it includes actors, directors, and genres
$\mathcal{R}$	Set of items and properties edge label	In the movies domain, it includes the edges "starring" for actors, "directed by" for directors, and "categorized" for genres
$\mathcal{U}$	Set of users	User that express preferences and require a recommendation
$\mathcal{D}$	Set of dialogues	Dataset of conversations. It is a set of conversations between a user and a recommender
$\mathcal{S}$	Sequence of user preferences	Entities (items and properties) properties mentioned by the user in the dialog
$\mathcal{V}(\int)$	Embedding operations	The linking of the element s of the sequence with its KGE

Table 1: Notations and definitions

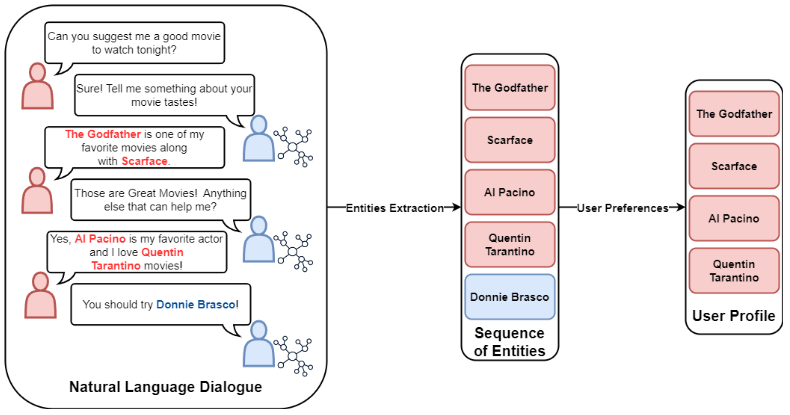

Can you suggest me a good movie to watch tonight?
Sure! Tell me something about your movie tastes!
The Godfather is one of my favorite movies along with Scarface.
Those are Great Movies! Anything else that can help me?
Yes, Al Pacino is my favorite actor and I love Quentin Tarantino movies!
You should try Donnie Brasco!
Natural Language Dialogue
Entities Extraction
The Godfather
Scarface
Al Pacino
Quentin Tarantino
Donnie Brasco
Sequence of Entities
User Preferences
The Godfather
Scarface
Al Pacino
Quentin Tarantino
User Profile

Figure 2: Processing conversational data to extract *sequences of entities* that represent user profile.

5.1 Description of the Architecture

As shown in Figure 3, the input of the model is represented by the *entity tokens* that are extracted from a dialogue. As previously stated, they represent the preferences of the user. Next, the input is processed through the KG embedding process we previously described and then passed through a Transformer architecture composed of an *embedding* layer, N *self-attention* layers, and an *output* layer. In the following, we describe the structure of the architecture.

Embedding Layer. The embedding layer is where our model significantly differs from other transformer-based recommendation models, such as TSCR [51]. Indeed, this layer uses the KGEs that are previously learned to provide the Transformer with the exogenous knowledge regarding the semantic connections between items and properties encoded in the KG. Moreover, in order to also encode information about the order in which the entities are mentioned in the original sequence, we employ *positional embeddings*. These embeddings are vector representations that encode the relative or absolute position of an element. This ensures that entities in the same position across different sequences share the same positional encoding. Formally, given a sequence S of entities e_k, we encode each entity at position k as the sum of two embeddings: the frozen pre-trained KGE $\mathbf{V(e_k)}$ and the positional embedding $\mathbf{p_k}$:

$$\mathbf{i_k^0} = \mathbf{V(e_k)} + \mathbf{p_k} \tag{2}$$

Next, we stack all entity representations to obtain the representation of the sequence. Where needed, we pad the sequence to the maximum sequence length to obtain the matrix $\mathbf{S^0}$. Next, at each self-attention layer n, we calculate an updated matrix $\mathbf{S^n}$. It it worth noting that in this process the KGEs remain frozen and are not modified during model training, ensuring that the model relies on the original graph-based representations.

Self-Attention Layer. A self-attention layer is composed of two distinctive sub-layers, namely a multi-head self-attention sub-layer and a Position-wise Feed-Forward Network (PFFN). The incorporation of these layers, as pointed out in [43], empowers the model with the capability to effectively capture *inter-dependencies* within sequences.

$$\mathbf{E^{n+1}} = \text{MultiHead}(\text{PFFN}(\mathbf{E^n})) \tag{3}$$

The PFFN is a network made up of a series of parallel and identical Feed-Forward Networks (FFN) with GELU activation. Each FFN is applied to each element in the sequence and aims to learn a representation of the element:

$$\text{PFFN}(\mathbf{E^n}) = [\text{FFN}(\mathbf{e_1^n})^T; \ldots; \text{FFN}(\mathbf{e_k^n})^T]^T \tag{4}$$

Finally, to ensure generality, mitigate overfitting, and speed up the training process, the output of each sub-layer (i.e., PFFN or multi-head) is normalized and subjected to dropout:

$$\text{LayerNorm}(\mathbf{E^n} + \text{Dropout}(\text{sublayer}(\mathbf{E^n}))) \tag{5}$$

Ouput Layer. After N self-attention layers, we obtain the final output matrix $\mathbf{E^N}$. Assuming that the masked element is in position $k+1$, we use the $\mathbf{e_{k+1}^N}$ to predict the item e_{k+1}. In order to achieve

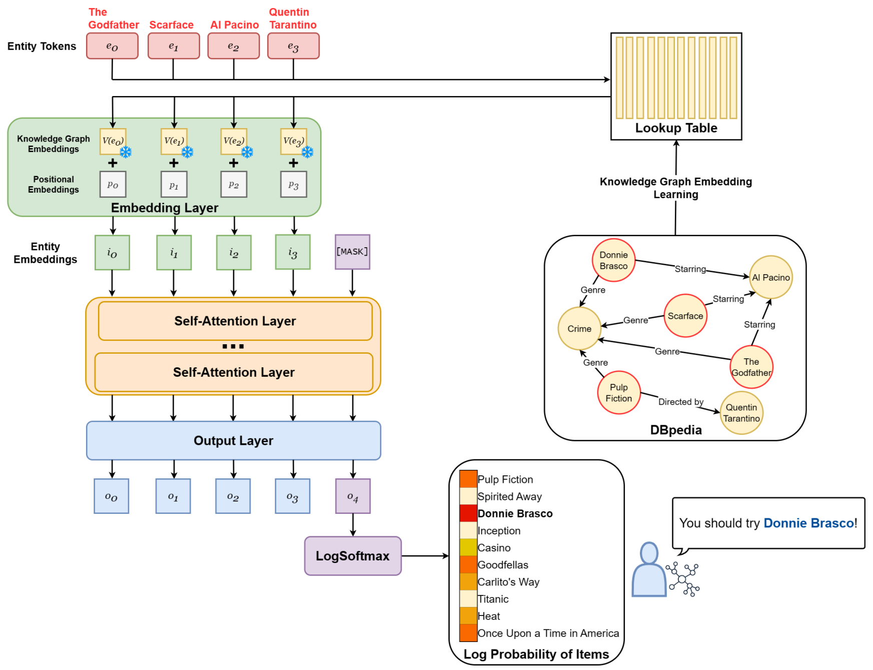


Figure 3: Each entity in a sequence of entities is encoded as the sum of the correspondent frozen KGE and positional embedding. To make a recommendation, a sequence with the special "[MASK]" token placed at the end is provided to the model, that can predict the item that more likely occupied the masked position.

this, we project $\mathbf{e}^{\mathrm{N}}_{\mathrm{k+1}}$ in the entities space with a two-layer feed-forward network with GELU activation in between:

$$\mathbf{o}_{\mathrm{k+1}} = \mathrm{GELU}(\mathbf{e}^{\mathrm{N}}_{\mathrm{k+1}}\mathbf{W}^{\mathrm{T}} + \mathbf{b}^{\mathrm{P}})\mathbf{E}^{\mathrm{T}} + \mathbf{b}^{\mathrm{O}} \tag{6}$$

Where $\mathbf{W} \in \mathbb{R}^{|I \cup F| \times d}$ and $\mathbf{E}^{\mathrm{T}} \in \mathbb{R}^{|I \cup F| \times |I \cup F|}$ is a learnable projection matrix, and $\mathbf{b}^{\mathrm{P}}$ and $\mathbf{b}^{\mathrm{O}}$ are bias terms.

5.2 Model Training and Inference

The model is trained on sequences of entities extracted from dialogues between a user and a CRS. Training is carried out through Masked Language Modeling [12], also referred to as the *cloze* task [41]. Generally speaking, this task involves masking a proportion of items within each sequence using a special token (*e.g.,* "[MASK]"), and the model is tasked with predicting the correct items. During training of KASCRS, we limit masking to only *item* tokens. In this way, we allow the model focus the learning on the recommendation of relevant items. This follows the learning strategy implemented in [51]. To ensure that only items are recommended, the score of non-item entities is set to $-\infty$. Finally, we define the loss for each masked sequence as the negative log-likelihood of the masked targets:

$$\mathcal{L} = \frac{1}{|\mathcal{S}^{(\mathrm{mask})}|} \sum_{e_m \in \mathcal{S}^{(\mathrm{mask})}} -\log P(e_m = e_m^* | \mathcal{S}') \tag{7}$$

where $\mathcal{S}'$ is the masked version of the sequence $\mathcal{S}$, $\mathcal{S}^{(mask)}$ is the set of masked items in $\mathcal{S}'$. e_m and e_m^* are the masked item and the target item, respectively. This loss function helps the model to maximize the ground truth items probabilities while simultaneously minimizing the competing items probabilities.

During testing, following the same protocol proposed in [51], for each target item (*i.e.,* each item mentioned by the recommender, shown in blue in Fig. 2) a special testing sequence is constructed. This testing sequence consists of the original sequence of entities extracted from the messages of user (shown in red in Fig. 2) truncated of the target item. Subsequently, the last entity in the sequence (*i.e.,* the target item) is masked. The resulting sequence, having the "[MASK]" token placed at the end, is then provided to the model, that predicts the item that more likely occupied the masked position of the "[MASK]" token. As shown in Fig. 3, this is done by applying a LogSoftmax to the output. This finally generates log probabilities for each item, indicating the model's likelihood of recommending it. Items are finally ranked based on this score, and the top-1 is returned as recommendation.

6 EXPERIMENTAL EVALUATION

To assess the effectiveness of our model, we evaluate the results w.r.t. two state-of-the-art datasets in the area of CRS. In particular, our experiments aimed to answer the following Research Questions (RQ):

- **RQ1 - Comparison to State of the Art**: How does our approach perform w.r.t. state-of-the-art CRS baselines?
- **RQ2 - Ablation Tests**: How do different components of the architecture (positional embeddings and KGE) and different types of properties (*i.e.,* actors, directors, genres) contribute to the overall performance?
- **RQ3 - Sensitivity Analysis**: What is the relative impact of mask probability and embedding dimensionality on the performance of KASCRS, and how do different KGE techniques impact the predictive accuracy?

6.1 Experimental Setting

Datasets. To conduct our experiments, we employ the ReDial dataset [24] and INSPIRED [19]. The ReDial corpus encompasses 11,348 dialogues, while INSPIRED comprises 1,001 dialogues. As stated in Section 3, starting from the dialogues in the datasets, we extract sequences of user preferences made up of items and properties. To ensure a fair comparison with the baseline models, we utilize the pre-processed dataset provided by CRSLab [48].
To construct the knowledge graph, different strategies were adopted. As for the ReDial dataset, actors and directors were extracted from DBpedia [2] and the genre properties were obtained by matching the DBpedia name with the MovieLens dataset [18]. Conversely, the properties for INSPIRED have been extracted from the metadata files released with the dataset[1]. Information about the characteristics of the knowledge graphs and about the length of the sequences for each datasets are summarized in Table 2.

Dataset	ReDial	INSPIRED
# Movies	6071	16733
# Actors	7164	32195
# Genres	21	27
# Directors	2807	9962
Avg. properties per movie	6.85	7.90
Avg. sequence length	7.24	12.88
Max. sequence length	43	47

Table 2: Comparison of ReDial and INSPIRED dataset statistics.

Protocol and Evaluation Metrics. For both the datasets, we exploited the original 8:1:1 splits for both datasets. To evaluate the effectiveness of our approach, we employ Recall@k. As previously stated, to calculate the metric we ask our model to predict the $k + 1$th entity (*i.e.,* the recommended item) based on the previous k entities that occur in the sequence. In particular, Recall@k assesses whether the ground truth item (*i.e.,* the actual recommendation proposed at the end of the dialogue), occurs in the top-k items returned by our recommender system. As ground truth, we used the items mentioned in the dialogue by the recommender.

To assess the significance in terms of Recall@k among the different configurations, we used McNemar test. The test is designed for *paired binary data*, which is the type of data we have because there is only one relevant item per test sequence. The test specifically looks for cases where one model succeeds while the other fails. If we reject the null hypothesis, it means that the models have different rates of success when trained on the same data [13].
Implementation Details. KASCRS is based on Pytorch. We run our experiments using an NVIDIA Tesla T4 GPU. Source code of the model is available in our repository[2], together with the data needed to reproduce the results of the experiments. As for For KGE techniques, we used the implementation of CompGCN available in Pykeen [1].
Model Parameters. Next, as regarding the architecture of KASCRS, the best results have been achieved with 2 layers and 2 attention heads. The maximum sequence length for input data is set at 50 so that no test sequence is cut in both datasets. The proposed model is trained using the stochastic gradient descent (SGD) optimizer with a batch size of 256, and hyperparameters specifically tailored to each dataset. For the INSPIRED dataset, the optimal parameters were found to be a learning rate of $5e - 4$ over 50 epochs, weight decay of 2, and dropout probability of 0.3. In contrast, the ReDial dataset required a higher learning rate of $5e - 3$ over 100 epochs, weight decay of 5, and dropout probability of 0.5. This difference in hyperparameters is attributed to the distinct item distribution patterns between the two datasets. The INSPIRED dataset exhibits a more balanced item distribution, while ReDial features a more imbalanced distribution. This imbalance necessitates stronger regularization techniques, such as higher weight decay and dropout, to prevent the model from overfitting to the prevalent items.
Sensitivity Analysis. To investigate the influence of various KGE algorithms on the overall performance of KASCRS, we compared CompGCN to three graph embedding techniques: *(1) RGCN* [36]: A relational graph convolutional network that explicitly models the relationships between nodes in a graph. We tested both 1-hop and 2-hop variants; *(2) TransE* [5]: A knowledge graph embedding method that represents entities and relations as points in a vector space. TransE seeks to minimize the distance between the representations of an entity and its corresponding relations; *(3) TransH* [46]: An extension of TransE to hyperbolic space. This enables TransH to model long-range dependencies between entities. Moreover, we also assessed the effectiveness of the model on varying of the graph embedding size (16, 32, 64, 128) as well as on different values of mask probability. Results are discussed in Section 6.2.
Baselines. To evaluate the effectiveness of our approach, we compare it to several state-of-the-art baselines, such as:

- *GPT-2* [34]: An auto-regressive pre-trained language model (PLM) that processes the utterances in a conversation as a single input sequence. The representation of the final token is used for recommendation generation;
- *BERT* [12]: PLM pre-trained using the MLM task on a large, general-purpose corpus. For recommendation generation, we extract the representation of the "[CLS]" token;
- *ReDial* [24]: An auto-encoder-based model based on HRED, introduced alongside the ReDial dataset;
- *KBRD* [9]: A Knowledge-based CRS model that leverages DBpedia to enhance items representation;
- *KGSF* [49]: A model that integrates DBpedia and ConceptNet [37] to enhance items and words representation;
- *UniCRS* [45]: A model that uses knowledge-enhanced prompt learning to train a generative model that is fed with pre-trained entity representations and task-specific prompts;
- *TSCR* [51]: Transformer-based CRS without external knowledge.

The implementations for GPT-2, BERT, ReDial, KBRD, and KGSF baselines are available in the CRSLab Toolkit[3], while UniCRS implementation is available in the repository provided in the paper[4]. For TSCR, since the authors did not provide any implementation, we developed the model based on the description in the paper.

[1]All the source graphs that can be used to obtain KGE for both the datasets are released in our GitHub repository, which is mentioned next.
[2]https://github.com/petruzzellialessandro/UMAP2024
[3]https://github.com/RUCAIBox/CRSLab
[4]https://github.com/RUCAIBox/UniCRS

Dataset	ReDial			INSPIRED		
Model/Recall	**Recall@1**	**Recall@10**	**Recall@50**	**Recall@1**	**Recall@10**	**Recall@50**
GPT-2	0.023	0.147	0.327	0.034	0.112	0.278
BERT	0.030	0.156	0.357	0.044	0.179	0.328
ReDial	0.023	0.129	0.287	0.003	0.117	0.285
KBRD	0.030	0.164	0.338	0.058	0.146	0.207
KGSF	0.039	0.183	0.378	0.058	0.165	0.256
UniCRS	0.051	0.224	0.428	0.094	0.250	0.410
TSCR	0.068	0.299	0.398	0.132	0.391	0.467
KASCRS	**0.121***	**0.409***	**0.759***	**0.161**	**0.448**	**0.648***

Table 3: Recommendation performances of our model and baselines . $^{(*)}$ indicates a significant improvement upon the best baseline in McNemar test with p-value < 0.01. Best performance are in bold.

6.2 Discussion of the Results

RQ1: Comparison to State of the Art. In order to answer RQ1, we compared our approach to other CRS baselines. The results are provided in Table 3. As shown in the table, KASCRS outfperforms all the baselines on both the datasets and on all the metrics we considered. Gaps with respect to the best-performing baseline (*i.e.*, TSCR) are statistically significant on all the metrics on ReDial, and on Recall@50 on INSPIRED.

As regards the other techniques, the results show that the introduction of external knowledge, as discussed in Section 2, significantly improves the performance. This is confirmed by the impressive gain achieved by KBRD, KGSF, and UniCRS with respect to the models that don't leverage the exogenous knowledge such as ReDial, BERT and GPT-2. Indeed, UniCRS outperformed BERT on the two datasets by 70% and 113%, respectively in terms of Recall@1. With respect to these methods, KASCRS obtained a significant improvement. As an example, compared to UniCRS (the best baseline that leverages external knowledge), the improvement on Recall@1 is over 135% on both the datasets.

As previously stated, the overall best-performing baseline on all the datasets is TSCR. This is in line with the results already shown in literature, and confirms that the introduction of transformer architecture in CRSs brought an additional improvement in the modeling of the preferences as a *sequence*. However, by comparing our results to TSCR we obtain a significant improvement of 77% on the ReDial dataset and 21% on the INSPIRED one. These results definitely confirm the intuitions behind this work, since we showed that the combined use of Transformers and Knowledge Graphs allows to better model sequences that encode preferences and needs of the users in a conversational recommendation setting.

RQ2: Ablation Tests. To answer RQ2, we carried out two different ablation tests. The first ablation study investigated the impact of the two main components of the architecture, that is to say, pre-trained embeddings based on knowledge graphs and the use of positional embeddings in the Transformer architecture described in Section 5.

As shown in in Table 4, the results validate our initial hypothesis, demonstrating that models that do not exploit KGE consistently perform worse than our configuration. This finding is particularly evident in the INSPIRED dataset, where all evaluation metrics are impacted by the absence of KGE. Similarly, in order to assess the contribution of sequential modeling, we conducted experiments by removing positional embeddings (POS) that are summed to the KGE, as depicted in Figure 3. The results on both datasets suggest that positional embeddings play a crucial role in capturing sequential dependencies between entities and relations. Specifically, the removal of POS led to a noticeable drop in performance across all metrics. To sum up, this test confirmed the intuitions and the design choices behind this work, since we showed that both the combined use of positional embeddings and pre-trained knowledge graph embeddings is fundamental to better encode user preferences in a conversational recommendation setting.

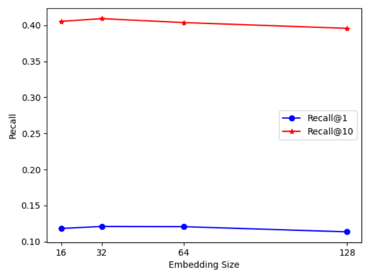
Next, in the second test we focused our attention on the characteristics of the KG. In particular, we compared the performance obtained on varying of different combinations of the properties available. Given that three type of properties are included in our KG (*i.e.,* actors, directors and genres), we carried out seven tests, one for each subset of properties. By referring to the toy KG presented in Fig. 1-c, each configuration is obtained by learning the KGE by maintaining just the edges labeled with a particular property, and by dropping all the others. As for the pre-processing of the sequences, in each test we maintained in the sequence only the properties that belong to the subset considered in the specific experiment. As an example, by referring to Figure 2, when the ablation test based on the *Directors* configuration is run, the property *Al Pacino* was excluded from the sequence, while *Quentin Tarantino* was maintained. All the tests were conducted on the best configuration of the model optimized per dataset, as previously described.

Dataset	ReDial			INSPIRED		
Model/Recall	**Recall@1**	**Recall@10**	**Recall@50**	**Recall@1**	**Recall@10**	**Recall@50**
KASCRS	**0.121**	**0.409**	**0.759**	**0.161**	**0.448**	**0.648**
w/o KGE	0.117	0.409	0.757	0.148	0.396	0.546
w/o POS	0.109	0.402	0.708	0.144	0.319	0.416

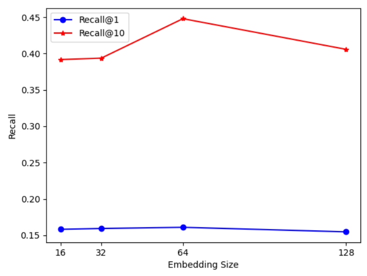
Table 4: Results for RQ2. Recall@k Score w/o Knowledge Graph Embedding and w/o Positional Embedding

As shown in Table 5, these test confirmed the effectiveness of our choices, since the exploitation of all the properties led to the best results on both the datasets. As regards the behavior of the single properties, "Actors" provides the best performance, in particular in terms of Recall@1. This remarkable outcome can be attributed to the characteristics and the topology of the knowledge graph. Indeed, as shown in Table 2, the number of "Actors" properties is the highest one, so it is likely that this group of properties may be more useful and discriminant to better model users preferences. However, even if the results obtained by "Directors" and "Genres" alone are lower, the results confirmed that their combination with the other properties available in the graph leads to the best results. This further confirms the validity of our approach, supporting the idea of exploiting KGs to encode exogenous knowledge about items and properties.

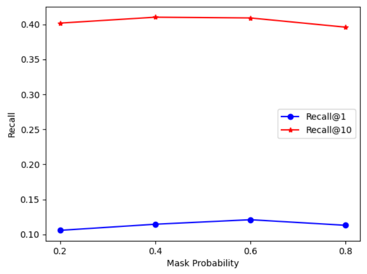
RQ3: Sensitivity Analysis.



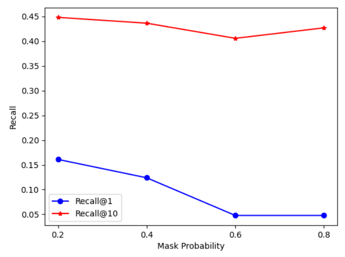
(a) Different embeddings size on ReDial



(b) Different embeddings size on INSPIRED



(c) Different mask probability on ReDial



(d) Different mask probability on INSPIRED

Figure 4: Results for RQ3. Recall@k Score of our sensitivity analysis

Dataset	ReDial			INSPIRED		
Properties	**Recall@1**	**Recall@10**	**Recall@50**	**Recall@1**	**Recall@10**	**Recall@50**
Actors + Directors + Genres (OUR)	**0.121**	**0.409**	**0.759**	**0.161**	**0.448**	0.648
Actors	0.117	0.403	0.751	0.159	0.392	**0.652**
Directors	0.094	0.403	0.750	0.116	0.400	**0.652**
Genres	0.104	0.403	0.754	0.116	0.427	0.638
Actors + Directors	**0.121**	0.406	0.754	0.159	0.402	0.644
Actors + Genres	0.117	0.408	0.754	0.159	0.420	0.646
Directors + Genres	0.114	0.407	0.754	0.108	0.427	0.640

Table 5: Results for RQ2. Recall@k Score for different property combinations

Dataset	ReDial			INSPIRED		
KGE algorithm/Recall	**Recall@1**	**Recall@10**	**Recall@50**	**Recall@1**	**Recall@10**	**Recall@50**
CompGCN - 2 layers (OUR)	**0.121**	**0.409**	**0.759**	**0.161**	**0.448**	0.648
CompGCN - 1 layer	**0.121**	0.400	0.746	0.159	0.420	0.648
RGCN 2 layers	0.116	0.408	0.727	0.159	0.386	0.646
RGCN 1 layer	0.104	0.403	0.714	0.158	0.420	**0.652**
TransH	0.106	0.403	0.717	0.159	0.420	0.644
TransE	0.113	0.405	0.747	0.159	0.420	0.644

Table 6: Results for RQ3. Recall@k Score of KASCRS with different KGE algorithms

Finally, we carried out a sensitivity analysis to investigate the influence of various and KGE algorithms on the overall performance of KASCRS. This is a fundamental analysis, aiming at showing that CompGCN is the technique able to better catch the characteristics of our KG. As shown in in Table 6, the results confirmed our choice since the best results were achieved using CompGCN with 2-hop embeddings. Similar results were achieved using the same algorithm with 1-hop embedding. This indicates that CompGCN is the most appropriate technique for this task.

Moreover, we also conducted a sensitivity analysis on two hyper-parameters of the overall architecture, such as the size of the embeddings and mask probability. To assess the impact of embedding size, we trained CompGCN with 2 layers using embedding of sizes 16, 32, 64, and 128. Figure 4 shows the trends of the performance on both datasets for different embedding sizes. To ensure the readability of results, we plot only Recall@1 and Recall@10, since Recall@50 follows the same trend. The figures 4a and 4b demonstrate that the two models achieve their best performances at different embedding sizes on the ReDial and INSPIRED datasets, respectively. This behavior can be attributed to the characteristics of the underlying knowledge graphs. As shown in Table 2, the ReDial dataset has fewer entities. This suggests that a smaller embedding size is sufficient to capture the relationships between them and to encode user preferences in a more effective way. Conversely, the use of a larger embedding size probably leads to noisy representations and lower overall performance. In contrast, the INSPIRED dataset has more entities, and this makes necessary the adoption of a larger embedding size to adequately represent the complex relationships between the elements in the KG. This is demonstrated by the superior performance of CompGCN on the INSPIRED dataset when the embedding size is set to 64.

Our experiments also revealed a difference in the optimal mask probability for achieving peak performance across the ReDial and INSPIRED datasets. While ReDial exhibited its best results with a mask probability of 0.6, INSPIRED achieved its peak performance at 0.2. This difference can be attributed to the distinct item distributions within each dataset. ReDial is characterized by shorter sequences and a tendency to present the same items repeatedly, thus exhibiting a strong popularity bias. To counteract this bias and encourage the model to learn representations of diverse items, a higher mask probability, such as 0.6, is the optimal choice. In this way, the model learns to predict a broader range of items and develops a more comprehensive understanding of their characteristics. In contrast, INSPIRED, presenting a more balanced item distribution, can effectively learn item representations with a lower mask probability, such as 0.2. This suggests that the model is less susceptible to popularity bias and can effectively capture item relationships even with fewer masked instances.

Overall, this analysis showed that the characteristics of the datasets should be carefully taken into account when the recommendation module of a sequential CRSs has to be trained. However, as we previously showed in the experiments, a proper choice of such parameters led our model to obtain results that overcome the current state of the art.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel approach to incorporate external knowledge into a transformer-based sequential recommendation system by utilizing graph embeddings to generate pre-trained representations of items and properties. Our approach integrates these item property embeddings alongside positional embeddings within the transformer architecture. This augmented representation allows the model to capture richer contextual cues, leading to improved recommendation precision as demonstrated by our experiments. We conduct ablation studies to investigate the impact of different KG information types and architectural components on the system's performance. Our results reveal that incorporating item properties information along with positional embeddings significantly enhances the system's ability to capture contextual relationships and recommend relevant items. These findings establish the value of incorporating external knowledge into CRSs and demonstrate the potential of our approach to enhance recommendation accuracy.

As future work, we will explore the possibility of integrating the system into a conversational module. This would allow us to leverage the system's ability to process and generate natural language to create more engaging and interactive dialogue experiences. Additionally, we could integrate further external knowledge from diverse sources, such as pre-trained word embeddings and distributional semantics models [30]. This would further enrich the learning process with new and diverse information, leading to a more accurate and comprehensive conversational recommendation model.

ACKNOWLEDGMENTS

We acknowledge the support of the PNRR project FAIR - Future AI Research (PE00000013), Spoke 6 - Symbiotic AI (CUP H97G22000210007) under the NRRP MUR program funded by the NextGenerationEU and project PHaSE (CUP H53D23003530006) - Promoting Healthy and Sustainable Eating through Interactive and Explainable AI Methods, funded by MUR under the PRIN 2022 program. The research has been partially supported by UNIBA-MAML

REFERENCES

[1] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. 2021. PyKEEN 1.0: a python library for training and evaluating knowledge graph embeddings. *The Journal of Machine Learning Research* 22, 1 (2021), 3723–3728.
[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web*, Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 722–735.
[3] Giacomo Balloccu, Ludovico Boratto, Gianni Fenu, Francesca Maridina Malloci, and Mirko Marras. 2024. Explainable Recommender Systems with Knowledge Graphs and Language Models. In *European Conference on Information Retrieval*. Springer, 352–357.
[4] Giacomo Balloccu, Ludovico Boratto, Gianni Fenu, and Mirko Marras. 2022. Post processing recommender systems with knowledge graphs for recency, popularity, and diversity of explanations. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 646–656.
[5] 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), 2787–2795.
[6] Robin Burke. 1999. The wasabi personal shopper: A case-based recommender system. In *AAAI/IAAI*. 844–849.
[7] Robin D Burke, Kristian J Hammond, and BC Yound. 1997. The FindMe approach to assisted browsing. *IEEE Expert* 12, 4 (1997), 32–40.
[8] Li Chen and Pearl Pu. 2007. Preference-based organization interfaces: aiding user critiques in recommender systems. In *User Modeling 2007: 11th International Conference, UM 2007, Corfu, Greece, July 25-29, 2007. Proceedings 11*. Springer, 77–86.
[9] Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang, and Jie Tang. 2019. Towards Knowledge-Based Recommender Dialog System. 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)*. Association for Computational Linguistics, Hong Kong, China, 1803–1813. https://doi.org/10.18653/v1/D19-1189
[10] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 815–824.
[11] Allegra De Filippo, Michele Lombardi, and Michela Milano. 2021. Integrated offline and online decision making under uncertainty. *Journal of Artificial Intelligence Research* 70 (2021), 77–117.
[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]
[13] Thomas G. Dietterich. 1998. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation* 10, 7 (10 1998), 1895–1923. https://doi.org/10.1162/089976698300017197 arXiv:https://direct.mit.edu/neco/article-pdf/10/7/1895/814002/089976698300017197.pdf
[14] Hendrik Drachsler, Hans Hummel, and Rob Koper. 2007. Recommendations for learners are different: Applying memory-based recommender system techniques to lifelong learning.
[15] Zuohui Fu, Yikun Xian, Yaxin Zhu, Yongfeng Zhang, and Gerard de Melo. 2020. Cookie: A dataset for conversational recommendation over knowledge graphs in e-commerce. *arXiv preprint arXiv:2008.09237* (2020).
[16] Chongming Gao, Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. 2021. Advances and challenges in conversational recommender systems: A survey. *AI Open* 2 (2021), 100–126. https://doi.org/10.1016/j.aiopen.2021.06.002
[17] Marco de Gemmis, Leo Iaquinta, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. 2011. Learning preference models in recommender

systems. *Preference Learning* (2011), 387–407.

[18] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (dec 2015), 19 pages. https://doi.org/10.1145/2827872

[19] Shirley Anugrah Hayati, Dongyeop Kang, Qingxiaoyang Zhu, Weiyan Shi, and Zhou Yu. 2020. INSPIRED: Toward Sociable Recommendation Dialog Systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 8142–8152. https://www.aclweb.org/anthology/2020.emnlp-main.654

[20] Dietmar Jannach. 2004. Advisor suite-a knowledge-based sales advisory system. In *ECAI*, Vol. 16. 720.

[21] Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, and Li Chen. 2021. A Survey on Conversational Recommender Systems. *ACM Comput. Surv.* 54, 5, Article 105 (may 2021), 36 pages. https://doi.org/10.1145/3453154

[22] Dietmar Jannach, Markus Zanker, Markus Jessenitschnig, and Oskar Seidler. 2007. Developing a conversational travel advisor with advisor suite. In *Information and Communication Technologies in Tourism 2007*. Springer, 43–52.

[23] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020. Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 304–312.

[24] Raymond Li, Samira Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards Deep Conversational Recommendations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 9748–9758.

[25] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. 2021. Empirical analysis of session-based recommendation algorithms: A comparison of neural and non-neural approaches. *User Modeling and User-Adapted Interaction* 31, 1 (2021), 149–181.

[26] Ahtsham Manzoor and Dietmar Jannach. 2022. Towards retrieval-based conversational recommendation. *Information Systems* 109 (2022), 102083. https://doi.org/10.1016/j.is.2022.102083

[27] Kevin McCarthy, James Reilly, Lorraine McGinty, and Barry Smyth. 2004. On the dynamic generation of compound critiques in conversational recommender systems. In *Adaptive Hypermedia and Adaptive Web-Based Systems: Third International Conference, AH 2004, Eindhoven, The Netherlands, August 23-26, 2004. Proceedings 3*. Springer, 176–184.

[28] Cataldo Musto, Pierpaolo Basile, Pasquale Lops, Marco De Gemmis, Giovanni Semeraro, et al. 2014. Linked Open Data-enabled Strategies for Top-N Recommendations.. In *CBRecSys@ RecSys*. 49–56.

[29] Cataldo Musto, Fedelucio Narducci, Marco De Gemmis, Pasquale Lops, Giovanni Semeraro, et al. 2009. STaR: a social tag recommender system. *Proceedings of the ECML/PKDD Discovery Challenge* (2009), 215–227.

[30] Cataldo Musto, Giovanni Semeraro, Pasquale Lops, and Marco de Gemmis. 2011. Random indexing and negative user preferences for enhancing content-based recommender systems. In *E-Commerce and Web Technologies: 12th International Conference, EC-Web 2011, Toulouse, France, August 30-September 1, 2011. Proceedings 12*. Springer, 270–281.

[31] Enrico Palumbo, Giuseppe Rizzo, Raphaël Troncy, Elena Baralis, Michele Osella, and Enrico Ferro. 2018. Translational models for item recommendation. In *European Semantic Web Conference*. Springer, 478–490.

[32] Marco Polignano, Cataldo Musto, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. 2021. Together is Better: Hybrid Recommendations Combining Graph Embeddings and Contextualized Word Representations. In *Fifteenth ACM Conference on Recommender Systems*. 187–198.

[33] Chanathip Pornprasit, Xin Liu, Natthawut Kertkeidkachorn, Kyoung-Sook Kim, Thanapon Noraset, and Suppawong Tuarob. 2020. Convcn: A cnn-based citation network embedding algorithm towards citation recommendation. In *Proceedings of the ACM/IEEE joint conference on digital libraries in 2020*. 433–436.

[34] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. https://api.semanticscholar.org/CorpusID:160025533

[35] Francesco Ricci and Quang Nhat Nguyen. 2007. Acquiring and revising preferences in a critique-based mobile recommender system. *IEEE Intelligent systems* 22, 3 (2007), 22–29.

[36] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2017. Modeling Relational Data with Graph Convolutional Networks. arXiv:1703.06103 [stat.ML]

[37] Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. *Proceedings of the AAAI Conference on Artificial Intelligence* 31, 1 (Feb. 2017). https://doi.org/10.1609/aaai.v31i1.11164

[38] Giuseppe Spillo, Cataldo Musto, Marco Polignano, Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. 2023. Combining Graph Neural Networks and Sentence Encoders for Knowledge-aware Recommendations. In *Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization, UMAP 2023, Limassol, Cyprus, June 26-29, 2023*. ACM, 1–12. https://doi.org/10.1145/3565472.3592965

[39] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (Beijing, China) *(CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 1441–1450. https://doi.org/10.1145/3357384.3357895

[40] Yueming Sun and Yi Zhang. 2018. Conversational recommender system. In *The 41st international acm sigir conference on research & development in information retrieval*. 235–244.

[41] Wilson L. Taylor. 1953. "Cloze Procedure": A New Tool for Measuring Readability. *Journalism & Mass Communication Quarterly* 30 (1953), 415 – 433.

[42] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2019. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082* (2019).

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[44] Meihong Wang, Linling Qiu, and Xiaoli Wang. 2021. A survey on knowledge graph embeddings for link prediction. *Symmetry* 13, 3 (2021), 485.

[45] Xiaolei Wang, Kun Zhou, Ji-Rong Wen, and Wayne Xin Zhao. 2022. Towards Unified Conversational Recommender Systems via Knowledge-Enhanced Prompt Learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 1929–1937. https://doi.org/10.1145/3534678.3539382

[46] 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.

[47] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[48] Kun Zhou, Xiaolei Wang, Yuanhang Zhou, Chenzhan Shang, Yuan Cheng, Wayne Xin Zhao, Yaliang Li, and Ji-Rong Wen. 2021. CRSLab: An Open-Source Toolkit for Building Conversational Recommender System. arXiv:2101.00939 [cs.CL]

[49] Kun Zhou, Wayne Xin Zhao, Shuqing Bian, Yuanhang Zhou, Ji-Rong Wen, and Jingsong Yu. 2020. Improving Conversational Recommender Systems via Knowledge Graph Based Semantic Fusion. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 1006–1014. https://doi.org/10.1145/3394486.3403143

[50] Kun Zhou, Wayne Xin Zhao, Hui Wang, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. Leveraging historical interaction data for improving conversational recommender system. In *Proceedings of the 29th ACM international conference on information & knowledge management*. 2349–2352.

[51] Jie Zou, Evangelos Kanoulas, Pengjie Ren, Zhaochun Ren, Aixin Sun, and Cheng Long. 2022. Improving Conversational Recommender Systems via Transformer-Based Sequential Modelling. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2319–2324. https://doi.org/10.1145/3477495.3531852